Data Efficient Multilingual Natural Language Processing

Xinyi Wang

CMU-LTI-22-014

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15123
www.lti.cs.cmu.edu

Thesis Committee:

Graham Neubig (Chair)	Carnegie Mellon University
Alan Black	Carnegie Mellon University
Yulia Tsvetkov	University of Washington, Seattle
Sebastian Ruder	Google Research

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Keywords: multilingual learning, cross-lingual generalization, machine translation

To my parents, Zixia Qin and Minxiang Wang.

Abstract

The adoption of neural network models has led to state-of-the-art performance in many NLP tasks on major languages that have large amounts of data (Devlin et al., 2019; Vaswani et al., 2017), but their improvements often lag behind for low-resource languages (Koehn and Knowles, 2017; Sennrich and Zhang, 2019a). This imbalance in NLP progress could lead to increasing disparities between people from different regions or in different socioeconomic conditions. The goal of this thesis is to develop methods which efficiently utilize the available data resources to build competitive NLP systems for all languages.

We focus on multilingual training, a particularly effective strategy for improving the model quality of low-resource languages while training parameter-efficient models (Zoph et al., 2016; Neubig and Hu, 2018; Devlin et al., 2019; Conneau et al., 2019). We identify three major challenges facing multilingual models. (1) The standard word embedding representation hinders the model's generalization to training signals across different languages, mainly because it does not have good inductive biases to account for the lexical similarities and discrepancies between different languages. (2) Searching for good multilingual data selection and balancing strategies requires multiple runs of model retraining because multilingual datasets are often highly imbalanced across different languages. (3) It is challenging to adapt a multilingual model to languages with very limited resources. To tackle the first two challenges for multilingual training, we propose better word representation methods for multilingual data that encourage positive transfer between languages, and design automatic methods to select and balance multilingual training data. To tackle the third challenge, we explore novel methods that adapt multilingual models to support language varieties that are often overlooked in existing multilingual benchmarks through model ensembling and data augmentation.

Acknowledgments

The PhD journey has been an extremely challenging and yet rewarding experience for me. I would not be able to reach the finish line without the help and love from the wonderful people in my life. Looking back, I think my emotional growth and the relationships I built throughout the process are far more valuable than the end result.

I cannot get to where I am without the guidance of my advisor, Graham Neubig. His work ethic and his breadth of knowledge in NLP research have been a constant inspiration for me. I am also extremely lucky to work with Yulia Tsvetkov, who encouraged me to tackle things that I didn't think were within my ability, and supported me when I almost lost confidence in myself. I also had the honor of working with Sebastian Ruder, who never hesitates to offer advice and who has helped me tremendously both on research and on career. I'm very lucky to have Alan Black as my committee member – his passion for research, especially for multilingual learning, has been essential in shaping my research interest. I want to thank my collaborators at LTI, in no particular order: Paul Michel, Hieu Pham, Junxian He, Junjie Hu, Pengcheng Yin, Zi-Yi Dou, Mengzhou Xia, Philip Arthur, Antonios Anasopoulos, Jamie Carbonell. I want to thank the members of NeuLab: I learned so much from everyone. In no particular order, I'd like to thank: Danish Pruthi, Craig Stewart, John Wieting, Hiroaki Hayashi, Aditi Chaudhary, Shuyan Zhou, Hao Zhu, Chunting Zhou, Shruti Rijhwani, Emmy Liu, Perez Ogayo, Lucio Dery, Patrick Fernandes, Kayo Yin and many others. I also want to thank my undergrad advisor David Chiang at Notre Dame, whose passion for teaching and research ignited my interest in NLP. I cannot get to where I am without the friends from CMU and Notre Dame, who have supported me through the ups and downs: Patricia Kay, Annette Sayre, Yang Zhang, Rose Doerfler, Jon Francis, Sanket Vaibhav Mehta, Jimin Sun, Daniel Spokoyny, Volkan Cirik, Joan Zhou, Sachin Kumar, Anjalie Field and many others. Finally, I would like to thank Neil Xu for helping me become a better version of myself.

I would like to thank my family for being the strongest backbone in my life, accepting me for who I am, and loving me without any constraints. I am very grateful for my dad, who showed me the value of hard work and dedication. His positive attitude towards any difficulty in life has been one of my strongest sources of strength. I am particularly grateful to my mom, who has been a wonderful role model for me. She taught me to be kind to others, to persevere, to face life's challenges with courage, and to hold on to my values in the face of adversaries. This thesis is dedicated to you both.

Contents

Chapter 1

Introduction

Human languages play an important role in human society. People use languages to communicate with each other, build a sense of community, and create diverse cultural heritages. There are over 7000 languages in the world, and each language is unique in its own way. Developing competitive natural language processing (NLP) tools for all languages in the world is crucial for the advancements in NLP to equitably benefit people across the globe.

While languages in the world have rich linguistic diversity, recent progress in NLP research mostly focuses on a few major languages, such as English, where a large amount of data is available. Deep learning methods and neural network based models have become the standard backbone for most NLP models, leading to state-of-the-art performance on a variety of NLP tasks such as text classification (Devlin et al., 2019), language modeling and generation (Radford et al., 2018; Brown et al., 2020), and neural machine translation (NMT) (Vaswani et al., 2017; Edunov et al., 2018). Because these models obtain significantly better performance with increasing parameter count and training data (Kaplan et al., 2020), researchers tend to focus on languages with a large amount of easily accessible data resources to push the frontiers of NLP research.

Despite the impressive progress in NLP brought about by deep learning, these advancements are not equally observed for languages with less digitized data for model training (Joshi et al., 2020). We refer to these languages with fewer resources as "low-resource languages". In fact, most languages in the world are low-resource languages; the top five languages in CommonCrawl, a popular source of web-crawled data for the development of NLP models, account for about 70% of the total data, while the majority of languages in the world have much less data or no data at all in CommonCrawl[1]. Low-resource languages also include regional language variations or dialects that are often overlooked in standard NLP datasets. For example, neural network models can reach impressive performance on English benchmarks, but their performance deteriorates on African American Vernacular English (Groenwold et al., 2020; Tan et al., 2020). Developing competitive models for low-resource languages is challenging mainly because the performance of neural models often lags behind the model performance on major high-resource languages when the amount of training data is scarce (Koehn and Knowles, 2017; Sennrich and Zhang,

[1] https://commoncrawl.github.io/cc-crawl-statistics/plots/languages

2019a). The imbalance in advancement of NLP technologies can lead to unfairness in allocating any potential benefits of the technologies to speakers of low-resource languages (Joshi et al., 2020; Blasi et al., 2022).

1.1 Research Motivations and Objectives

This thesis aims to address the challenges facing low-resource languages by designing intelligent strategies to use data when the available resources for these languages are limited. The works in this thesis are inspired by two general attributes of human languages. The first attribute is that languages around the world share many **commonalities**. For example, Galician, a European language used by more than 2 million people, shares many lexical similarities to Portuguese and Spanish. The similarities between different languages allow us to utilize data from related languages to improve performance in low-resource languages. However, each language has its own **unique linguistic properties** that differentiates it from other highly related or even mutually intelligible varieties. Bhojpuri, for example, is very similar to a relatively high-resourced language Hindi, but it still has systematic differences in certain words and phrases. Therefore, it is essential to account for these language variations when developing competitive NLP systems for low-resource languages.

The first two parts of the thesis focus on multilingual training, or training a single model jointly on both high-resource and low-resource languages. Multilingual training is a particularly effective strategy to improve low-resource languages on many NLP tasks such as machine translation (Zoph et al., 2016; Neubig and Hu, 2018), text classification (Devlin et al., 2019; Conneau and Lample, 2019; Conneau et al., 2019) and question answering (Clark et al., 2020). However, it has two major challenges due to the heterogeneous training datasets from many different languages.

1. It is challenging to determine the amount of training data from each language to use while training jointly on all languages, because the amount of training data in each language is often very different.
2. The standard word embedding method in neural network models does not have an inductive bias to account for the lexical similarities and differences between languages, which limits positive transfer between different languages.

To tackle the two problems for multilingual training, the first part of the thesis outlines intelligent data selection and utilization strategies for multilingual data, and the second part aims to design better data representation methods for multilingual data.

Then, we turn to another important category of low-resource languages that have historically been overlooked in developing language technologies. The third part of the thesis focuses on adapting multilingual models to languages and language variations with very limited resources, such as languages with limited textual data or regional dialects. It is particularly challenging to build competitive NLP models for these languages because they often have very little or no data at all, and they generally have received relatively less attention from the NLP community. Nevertheless, improving NLP systems for these lan-

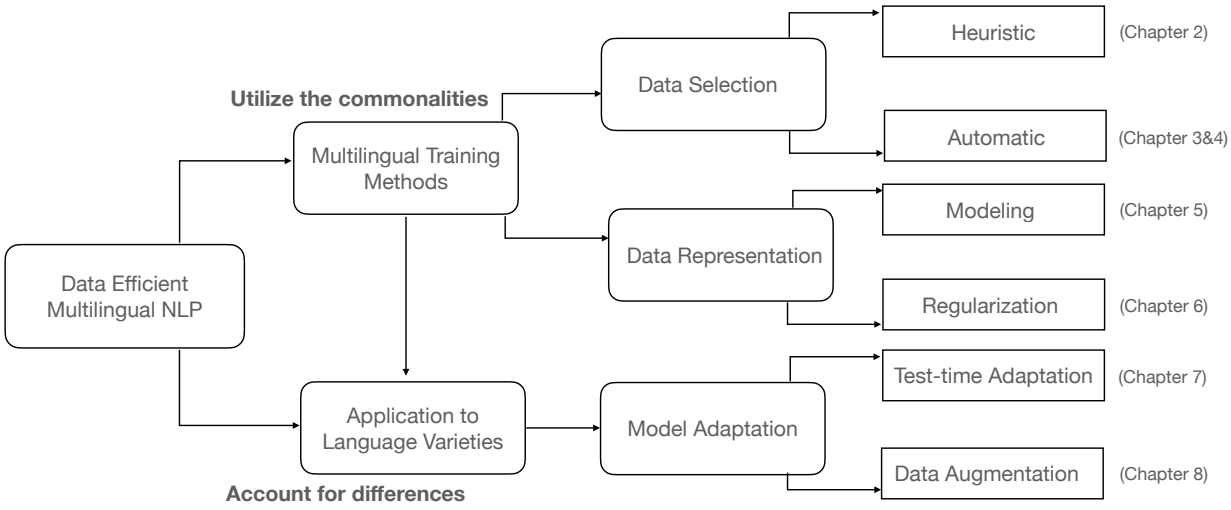

Figure 1.1: Content structure of the thesis.

guage varieties is an important step towards making inclusive NLP models that benefit all people.

1.2 Contributions

This theme of this thesis revolves around the commonalities and unique properties of each language. Fig. 1.1 illustrates the contributions of the proposal under this theme.

To leverage language similarities, we focus on multilingual training methods that encourage positive transfer between different languages. First, we observe that intelligent data sampling methods which consider the relatedness of different languages can have a significant positive impact on multilingual training (Chapter 3,Chapter 4,Chapter 5). Second, we find that improvements in model architecture that incorporate inductive bias about the commonalities between languages can significantly improve multilingual training (Chapter 6). Furthermore, accounting for the word segmentation differences between languages is crucial for cross-lingual transfer (Chapter 7).

Besides leveraging language commonalities, we also recognize the linguistic diversities of the worlds' languages and aim to adapt multilingual NLP models to speakers of language varieties overlooked in existing research and benchmarks. These language varieties often have no or very limited textual data for training. Therefore, we propose a novel algorithm to compose model parameters from related languages at test time for an unseen language variety (Chapter 8) and a general data augmentation method to synthesize data into these languages (Chapter 9).

1.3 Thesis Outline

A more detailed structure of the proposal is as follows:

- Part I investigates data selection and balancing strategies that maximize the utility of multilingual

training data.

  - Chapter 3 investigates heuristic methods to select data from many different languages to improve the NMT model's performance on a low-resource language. The work is published at ACL 2019 (Wang and Neubig, 2019).
  - Chapter 4 proposes a framework that automatically learns the optimal data usage strategy that maximizes the performance of the model in a validation set. We design an algorithm under the framework that optimizes the schedule for using available multilingual data to improve a given low-resource language. This work is published at ICML 2020 (Wang et al., 2019c).
  - Chapter 5 generalizes the algorithm in Chapter 4 to learn a data sampling schedule of multilingual training data to maximize the model performance on many different languages. This algorithm also allows one to have different optimization priority for multilingual objectives. This work is published in ACL 2020 (Wang et al., 2020a).

- Part II designs better multilingual word representations.
  - Chapter 6 proposes a novel word embedding method for multilingual data that decomposes the word embedding into a lexical embedding that reflects the spelling of the word, a latent semantic embedding, and a language-specific transformation. This design can encourage positive transfer from high-resource languages to low-resource languages because it maps similar words from different languages into closer vector representations. This work is published at ICLR 2019 (Wang et al., 2019b).
  - Chapter 7 focuses on improving the word representation for massive multilingual pretrained models for better cross-lingual transfer. In this work, we first apply existing subword regularization strategies during fine-tuning stage of the multilingual pretrained models to alleviate the word segmentation discrepancies between different languages. We then propose a method that further enforces the model prediction consistency under different word segmentation. This work is published at NAACL 2021 (Wang et al., 2021b).
- Part III focuses on model adaptation methods for challenges facing languages with limited resources and fine-grained variations.
  - Chapter 8 focuses on adapting pretrained multilingual models for unseen dialectal variations at test time. In this work, we first show that using an ensemble of related language adapters can lead to more robust performance for an unseen language than using any individual language adapter. We then propose a test time adaptation algorithm that adjusts the ensemble weights to minimize the entropy loss for each test input from the new language variety. This work is published at Findings of EMNLP 2021 (Wang et al., 2021c).
  - Chapter 9 focuses on adapting pretrained multilingual models to languages with little or no textual data, which account for over 80% of the 7000+ languages in the world. We propose a pipeline that leverages bilingual lexicons, an under-utilized data source with much better

language coverage than conventional data sources, to synthesize textual data into the low-resource languages. This work is published at ACL 2022 (Wang et al., 2022).

Chapter 2

Background

This chapter outlines the background and related work for the thesis. We first give an overview of low-resource languages (§ 2.1). Next, we briefly introduce the NLP applications covered in this thesis: neural machine translation (§ 2.2) and pretrained models (§ 2.3). Finally, we give an overview of multilingual training, which is the main methodology studied in this thesis (§ 2.4).

2.1 Low-resource Language Varieties

Since this thesis focuses on challenges for low-resource language varieties, we provide a concrete definition of low-resource languages and explain their characteristics. There are more than 7000 languages in the world. Although several major languages have abundant textual data to benefit from state-of-the-art deep learning methods, the majority of the world's language varieties fall behind by a large margin due to lack of resources. The languages that have inferior performance using standard modeling and training techniques due to limited labeled data, monolingual data, or other resources are often referred to as low-resource languages. Several previous works have aimed to study and categorize low-resource languages (Singh, 2008; Cieri et al., 2016; Magueresse et al., 2020). In particular, Joshi et al. (2020) study the current state of NLP progress by dividing all languages in the world into 6 groups based on the availability of labeled and unlabeled textual data. Although there is no standard definition or categorization of low-resource languages, it is helpful to have some understanding of these languages as a guideline for allocating future research efforts.

There are two axes to consider when defining low-resource languages. First, the categorization of low-resource languages depends on the particular NLP task that one wishes to solve. For example, there are abundant parallel data between English and French to train a good machine translation model with the standard settings. However, there are far fewer data available in French for other NLP tasks such as dialogue and question answering. Second, the state of technology development also has big impact on what languages are considered low-resource. This is because new NLP technologies could be developed such that it would require less data or make better use of new sources of data, making the performance of some low-resource languages defined under the old technology competitive enough to be removed

from the group. Therefore, the categorization of low-resource languages is a fluid concept that should be re-evaluated periodically to provide up-to-date guidance to the NLP community.

With these two principles in mind, we provide a rough sketch of low-resource languages based on the state of NLP technologies for machine translation at the time of writing this thesis. For simplicity of definition, we consider the number of monolingual and labeled *textual* data as a criterion to define low-resource languages. It is worth mentioning that some low-resource languages with very little textual data could have more data in other formats, such as video/audio recordings and linguistic information. For example, there are several hundred local languages spoken in Indonesia. Since many of them are mostly used colloquially in casual conversations, they tend to have very little written textual data while audio is a more natural data form for these languages (Aji et al., 2022). At the time of writing this thesis, the majority of the works on NLP and multilingual learning focus on text-only models, so we still use the availability of textual data as the criterion for categorizing low-resource languages. It is an important future direction to utilize alternative data sources such as speech and audio for low-resource languages.

There are three types of low-resource languages. 1) **No-Text**: these languages have virtually no labeled or unlabeled data, which account for about 80% of languages in the World (Joshi et al., 2020). Some examples of this type of languages are Popoloca, Bora, and Dahalo. Many languages within this category are endangered languages that are at risk of disappearance as their speakers shift to other languages or die out. 2) **No-Label**: these languages also have almost no labeled data, but they might have a very small amount of unlabeled data. Some examples of these languages are Greenlandic and Romani. Languages in this category often have a decent number of native speakers, but there is very limited attention to these languages when curating data sets and developing specialized NLP systems. 3) **Low-Label**: these languages have a small amount of labeled and unlabeled data, but the size of their data is still orders of magnitude lower than the major languages such as English. Some examples of these languages are Zulu, Cebuano, and Maltese. This group of languages are often consistently utilized by a small group of native speakers.

The works in this thesis cover all three groups of low-resource languages. Part I and Part II focus on Low-Label languages that have some amount of parallel data and monolingual data included in joint multilingual training. Part III focuses on methods that adapt multilingual models to the No-Text and Few-Label languages.

2.2 Neural Machine Translation

Machine translation is one of the longest standing NLP research problems. Early attempts to use computers to translate between human languages date back as far back as the 1950s (Hutchins, 2000, 2004). Statistical machine translation (SMT) became more popular as computers became more powerful and accessible. Progress in SMT enabled the development and commercialization of practical MT systems (Chiang, 2005; Koehn et al., 2007). More recently, neural network models have become the state-of-the-art backbone of MT systems (Bahdanau et al., 2015; Vaswani et al., 2017). Neural machine translation (NMT) generally achieves much better performance than SMT while having a simple end-to-end system archi-

tecture. In this section, we give a brief introduction to NMT systems.

Given a parallel corpus from the source language S to the target language T, an NMT model is optimized to translate a source sentence $x \in S$ into its corresponding target sentence $y \in T$. An NMT model generally has two components: an encoder and a decoder. The encoder produces a representation of the source sentence x:

$$h_x = f_{\text{enc}}\left(\text{emb}(x)\right) \tag{2.1}$$

where $\text{emb}(x)$ is the embedding of the input x. The decoder then computes the representation for each target timestep using h_x:

$$h_y^i = f_{\text{dec}}(h_x, \text{emb}(y^{t<i})) \tag{2.2}$$

where $\text{emb}(y^{t<i})$ is embedding of the target words before the current timestep. At every timestep i, the decoder can generate a word from the target vocabulary using the scores from $\text{softmax}(W \cdot h_y^i)$, where W is the linear projection matrix that maps the dense vector representation to the target vocabulary.

The details of the functions $f_{\text{enc}}(\cdot)$ and $f_{\text{dec}}(\cdot)$ depend on the choice of model architectures. We could use the Recurrent Neural Network (RNN; Hochreiter and Schmidhuber, 1997; Chung et al., 2014) or the transformer model (Vaswani et al., 2017). Although RNN and transformer have different advantages that make them suitable for different use cases, the state-of-the-art NMT systems are currently dominated by transformer models.

2.3 Deep Pretrained Models

Labeled data is crucial for training an NLP model to perform specific tasks, but it can be expensive to annotate the data for each task we wish to support. On the other hand, there is much more unlabeled text available, such as Wikipedia pages, news articles, and books. Many researchers have proposed to use unlabeled texts to directly improve the performance of NLP applications (Sennrich et al., 2016a; Ruder and Plank, 2018). More recently, a two-stage pretraining-fine-tuning paradigm has been shown to be particularly effective for neural network models (Devlin et al., 2019; Raffel et al., 2020). It involves learning a general language representation model on unlabeled text, which we can then use for various downstream NLP tasks. The goal is that by training the model on large quantities of unlabeled text, we can get a set of parameters that can be a good initialization point for any NLP task.

We often use the term pretraining to refer to the first step of training the language representation model on unlabeled text. There are generally three types of modeling choices for pretraining. The first type of models are decoders trained using auto-regressive language modeling objective (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018a), where the goal is to predict the next word in the sentence given the previous words. The second type of pretrained models are encoders, often trained using the masked language modeling objective (MLM), where the goal is to predict the words that are masked out in a sentence given the unmasked context (Devlin et al., 2019; Conneau and Lample, 2019; Conneau et al., 2019). The third type of pre-trained models is encoder-decoder models, similar to

the NMT model discussed in § 2.2. These models have more flexibility in using both monolingual and parallel data with different training objectives, such as reconstructing perturbed inputs or translating the input sentence into a different language (Raffel et al., 2020; Xue et al., 2021b; Lewis et al., 2020a). The pretraining step usually requires large amounts of unlabeled data and computational resources so that the model can learn a good representation for textual input.

After pretraining, the model can be finetuned for a particular downstream task using a relatively small amount of labeled data. The pretraining-fine-tuning paradigm is currently the state-of-the-art method for various NLP tasks such as classification, sequence tagging, and question answering. It is also a promising paradigm for supporting low-resource languages because pretrained models can reach competitive performance on many tasks with much less labeled data compared to training a model from scratch (Wu and Dredze, 2019; Brown et al., 2020)

2.4 Multilingual Training

The lack of both labeled and unlabeled data is the main difficulty of training competitive NLP systems for low-resource languages. Therefore, multilingual training has become a particularly effective approach for improving the performance of low-resource languages through cross-lingual transfer (Zoph et al., 2016; Neubig and Hu, 2018; Devlin et al., 2019; Conneau et al., 2019; Babu et al., 2021b). In this section, we give a brief definition of multilingual training, clarify its optimization objectives, and outline several challenges to applying this technique in practice.

2.4.1 Definition and applications

Given a multilingual dataset with N languages $D_{\text{multi}} = \{D_1, D_2, ..., D_N\}$ where each D_i contains data from a unique language or language pair, multilingual training optimizes the model parameter jointly on all the N datasets. Multilingual training has a rich history (Schultz and Waibel, 1998; Mimno et al., 2009; Shi et al., 2010; Täckström et al., 2013), but has become particularly prominent in recent years due to the ability of neural networks to easily perform multitask learning (Dong et al., 2015; Plank et al., 2016; Johnson et al., 2016b). Multilingual training can be applied to various NLP models and tasks. Here we mainly introduce its applications in NMT models and deep pretrained models because they are the models we focus on for the rest of the thesis.

Multilingual NMT In § 2.2 we explained that a monolingual NMT model that translates between a single language pair S-T is trained on parallel data from the source language S to the target language T. To train a multilingual NMT model, we jointly optimize the parameters of the NMT model on D_{multi}, where each $D_i \in D_{\text{multi}}$ contains parallel data from a source language S_i to a target language T_i. We usually assign a tag for each target language T_i. Let s_i-t_i be a parallel sentence from the training dataset D_i, we add the target language tag for T_i to s_i-t_i when we optimize the NMT model on this training example. Here is an example of the parallel training data where S is English and T is French:

- s_i: How are you ?

- t_i: <2fr> Comment ça va ?

At inference time, we can generate translations in any of the N target languages by feeding the corresponding target language tag to the model. Due to the usage of target language tags and joint training in many parallel datasets, multilingual models can translate between any source language S_i to any target language T_j, even if the multilingual training data D_{multi} do not contain a parallel dataset from S_i to T_j. This property is often referred to as zero-shot translation in existing works on multilingual NMT (Johnson et al., 2016b; Firat et al., 2016b). This attribute can significantly decrease the quadratic number of parallel datasets required to support translation between any two of the N languages in the training data.

Multilingual pretrained models As discussed in § 2.3, deep pretrained models are optimized on large amounts of unlabeled data during pretraining, and then they can be finetuned on labeled data in various NLP tasks. Similarly, multilingual pretrained models are first optimized on unlabeled multilingual dataset D_{multi} that contains unlabeled data in N different languages. To utilize the multilingual pretrained model for a downstream task, we can finetune the model on the labeled data for the task in a high-resource language, often English. The finetuned model can then be directly used to support other languages for the downstream task, although it is not finetuned on labeled data in these languages. This property of pretrained multilingual models is often called zero-shot cross-lingual transfer, because it allows the model to support low-resource languages without any labeled data in these languages (Wu and Dredze, 2019). It is a promising direction for supporting low-resource languages in NLP models because unlabeled data are more prevalent than labeled data and labeled data are often much more expensive to annotate. If we have labeled data in many different languages, we can also finetune the model jointly on the multilingual labeled data. This could lead to better performance on the downstream task for low-resource languages (Conneau et al., 2019; Hu et al., 2020).

As we can see, the dataset $D_i \in D_{\text{multi}}$ for multilingual NMT models contains parallel data from a source language S_i to a target language T_i, while for pretrained multilingual models, $D_i \in D_{\text{multi}}$ is data from a single language L_i. For simplicity of notation, we use L_i to represent the language or language pair for the dataset D_i.

2.4.2 Multilingual optimization objectives

There are generally two different optimization objectives for multilingual training. **The first objective** is to obtain the best possible model for a particular language L_i, often a low-resource language, by leveraging cross-lingual transfer through training the model on all or a subset of D_{multi}. Multilingual training can be considered as a form a data augmentation, because the model can use the additional data from the related languages to learn the linguistic patterns shared between the related languages and the low-resource language L_i. **The second objective** for multilingual training is to support all languages $\{L_1, L_2, .., L_N\}$ in the training data. A multilingual model trained using this objective is a much more parameter efficient method to support all N languages than training N different monolingual models. Generally, this type of multilingual model also performs better for low-resource languages compared to

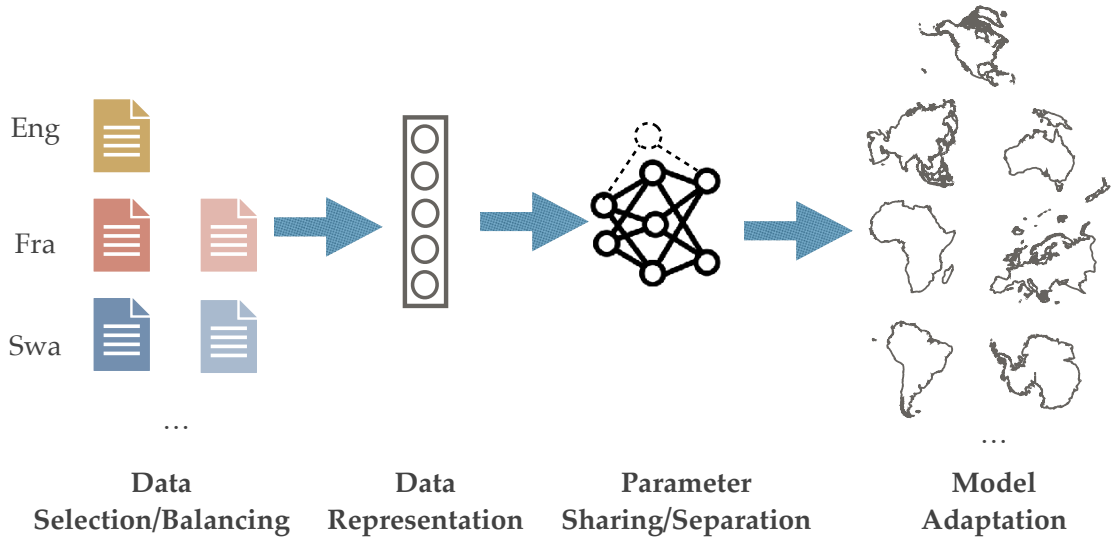

Figure 2.1: The four stages of multilingual training. The works in this thesis mainly focus on three of the four stages: data selection/balancing, data representation, and model adaptation.

monolingual models (Aharoni et al., 2019). For this objective, the standard evaluation criterion is often the average performance over all N languages (Hu et al., 2020; Conneau et al., 2022). While the average performance of the languages is a simple criterion to evaluate the general performance of multilingual models across all languages, it might not be a fair metric because it might bias towards high-resource languages that are better represented in the evaluation data. More recently, Blasi et al. (2022) question the fairness of the strategy and propose to consider other factors such as speaker population and economic impact to calculate the overall utility of a multilingual model.

The two multilingual optimization objectives may sometimes be conflicting, but are often interdependent. For example, the best performing model for a particular language L_i can often be obtained by first optimizing the model θ jointly on several languages related to L_i and then finetuning θ only on data in L_i (Neubig and Hu, 2018).

2.4.3 Challenges of multilingual training

We categorize the challenges of multilingual training based on four training stages shown in Fig. 2.1. In this section, we clarify the challenges in each training stage and introduce some existing work to tackle these challenges.

Data selection and balancing

Two important attributes of multilingual datasets make data selection and balancing extremely important for successful multilingual training. First, the size of each $D_i \in D_{\text{multi}}$ can have a very large variance, making multilingual datasets highly imbalanced. For example, the top five languages make up about 70% of the CommonCrawl corpus, while the remaining $160+$ languages only account for 30% of the

data[1]. Second, using training examples from D_i of a specific language or language pair can have very different impact on the model's performance on other languages. This is because different languages have very different levels of relatedness.

Various heuristic methods have been proposed to select data for different multilingual training objectives. To train a model for a specific language L_i, (Neubig and Hu, 2018; Lin et al., 2019) rely on linguistic knowledge and other heuristics to select a subset of the training languages that are most likely to benefit the performance of the model on L_i. Aharoni et al. (2019); Arivazhagan et al. (2019); Conneau et al. (2019) aim to support all N languages in the training data, so they rely on a heuristic data balancing schedule based on the size of each dataset scaled by a temperature term.

Data representation

It is challenging to design a good input representation for multilingual text because different languages can have different scripts or vocabulary. The standard method of representing an input word for an NLP model generally involves two steps: first, a heuristic word segmentation model learned using the training corpus is used to segment the word into commonly occurring subword units; second, each subword unit is mapped to a dense vector through an embedding table. To encode a diverse vocabulary from hundreds or even thousands of languages, multilingual models usually rely on standard subword segmentation methods such as byte-pair-encoding (BPE; Sennrich et al., 2016b) or unigram language models (Kudo and Richardson, 2018) to learn a single vocabulary shared by all languages. This method often leads to suboptimal segmentation and representation of multilingual text for two reasons: first, low-resource languages are often assigned smaller number of subword tokens than high-resource languages; second, the embedding lookup table simply converts words with slightly different spellings in different languages into completely separate vectors, limiting cross-lingual transfer.

Several works propose to optimize subword-sensitive word encoding methods for NLP models. The first line of work focuses on adding character information in the word embedding, either through convolutional neural networks (Kim, 2014; Lee et al., 2017; Cherry et al., 2018a; Kim et al., 2016; Józefowicz et al., 2016; Ma et al., 2020) or other character embedding composition methods (Ataman and Federico, 2018; Clark et al., 2022; Xue et al., 2021a). Apart from using character information, a different line of work aims to design better subword segmentation models by adding phrases in the vocabulary (Zhang and Li, 2020), learning separate vocabularies for each language cluster (Chung et al., 2020), or designing better learning methods for the subword segmentation model (He et al., 2020; Xu et al., 2021).

Parameter sharing and separation

While multilingual training generally optimizes all parameters in the model on all N languages, there are different design choices on whether or how to allocate a subset of the parameters for a specific language or language group. Intuitively, the parameters shared across languages maximize the learning of commonalities such as borrowed vocabularies, which could enable more effective cross-lingual transfer

[1] https://commoncrawl.github.io/cc-crawl-statistics/plots/languages

that improves the performance of low-resource languages. However, different languages have unique linguistic properties that might be more effectively modeled through language specific parameters.

There is no standard or uniformly best strategy for allocating language-specific parameters. Some early works on multilingual NMT only share specific components of the model across all languages while having language specific encoder, decoder, or attention (Dong et al., 2015; Firat et al., 2016a). (Ha et al., 2016; Johnson et al., 2016b) propose to share all NMT model components across different translation directions while adding a language embedding to the input sentence. While sharing all model parameters is a simple and effective approach, (Aharoni et al., 2019; Arivazhagan et al., 2019) show that increasing the number of languages for a multilingual NMT model could lead to lower performance, especially for high-resource languages, due to the limited model capacity. Several works aim to alleviate this problem by allocating more fine-grained language-specific parameters such as certain model layers or attention modules (Blackwood et al., 2018; Wang et al., 2019d; Zhu et al., 2020). Others propose to add a small amount of additional parameters during multilingual training (Zhang et al., 2021a; Wang et al., 2020c) or adaptation (Bapna et al., 2019; Lin et al., 2021). Mixture-of-expert models can directly increase model capacity while maintaining similar computation cost by only activating part of the model during multilingual training. Prior works show that the mixture-of-expert models could lead to better performance when scaling the multilingual model to a large number of languages (Shazeer et al., 2017; Kudugunta et al., 2021).

Adaptation

After we train a big multilingual model according to the objective of optimizing the performance of all N languages in D_{multi}, it is beneficial to adapt the model to a specific language L_i. That is, we can continue to optimize the model according to the objective of maximizing its performance on L_i, so that the speakers of L_i can enjoy the best performing model for their language. As large models are becoming essential for state-of-the-art performance (Shoeybi et al., 2019; Brown et al., 2020), effective model adaptation could maximize the utility of large pretrained models while minimizing computational resources, model storage, and model deployment efforts.

There are generally two goals for adapting large multilingual models: increasing parameter efficiency and maximizing final performance. Several works focus on parameter-efficient adaptation of large pre-trained models through adding a small amount of additional parameters for each individual language or task while keeping the large model fixed (Bapna et al., 2019; Pfeiffer et al., 2020b; Li and Liang, 2021; Guo et al., 2021; Üstün et al., 2020; Karimi Mahabadi et al., 2021). Other works focus mainly on maximizing the model performance for a particular language L_i, generally through continue training the model on the dataset D_i in L_i (Neubig and Hu, 2018; Wang et al., 2020b; Chau et al., 2020). Additionally, (Chau and Smith, 2021; Muller et al., 2021) explore several model adaptation strategies, including transliterating the data to Latin script and expanding the vocabulary during adaptation.

Summary

In this section we introduce the challenges in the four stages of multilingual training: data selection, data representation, parameter sharing/separation, and model adaptation. The works in the thesis are organized according to these four multilingual training stages. In particular, we focus mainly on three of the four challenges in data selection (Part I), data representation (Part II), and model adaptation (Part III). The next chapter will discuss our efforts to tackle challenges in data selection.

Part I

Data selection

Chapter 3

Target Conditioned Sampling: Optimizing Data Usage for Multilingual Neural Machine Translation

In order to explore data selection methods for multilingual training, we first focus on heuristic data selection strategies for NMT. To improve low-resource NMT with multilingual corpora, training on the most related high-resource language only is often more effective than using all data available (Neubig and Hu, 2018). However, it is possible that an intelligent data selection strategy can further improve low-resource NMT with data from other auxiliary languages. In this chapter, we seek to construct a sampling distribution over all multilingual data, so that it minimizes the training loss of the low-resource language.[1]

3.1 Introduction

Multilingual NMT has led to impressive gains in translation accuracy of low-resource languages (LRL) (Zoph et al., 2016; Firat et al., 2016a; Gu et al., 2018a; Neubig and Hu, 2018; Nguyen and Chiang, 2018). Many real world datasets provide sentences that are multi-parallel, with the same content in a variety of languages. Examples include TED (Qi et al., 2018), Europarl (Koehn, 2005), and many others (Tiedemann, 2012). These datasets open up the tantalizing prospect of training a system on many different languages to improve accuracy, but previous work has found methods that use only a single related (HRL) often out-perform systems trained on all available data (Neubig and Hu, 2018). In addition, because the resulting training corpus is smaller, using a single language is also substantially faster to train, speeding experimental cycles (Neubig and Hu, 2018). In this chapter, we go a step further and ask the question: can we design an intelligent data selection strategy that allows us to choose the most relevant multilingual data to further boost NMT performance and training speed for LRLs?

[1]The code can be found at https://github.com/cindyxinyiwang/TCS.

Prior work has examined data selection from the view of domain adaptation, selecting good training data from out-of-domain text to improve in-domain performance. In general, these methods select data that score above a preset threshold according to some metric, such as the difference between in-domain and out-of-domain language models (Axelrod et al., 2011; Moore and Lewis, 2010) or sentence embedding similarity (Wang et al., 2017). Other works use all the data but weight training instances by domain similarity (Chen et al., 2017), or sample subsets of training data at each epoch (van der Wees et al., 2017a). However, none of these methods are trivially applicable to multilingual parallel datasets, which usually contain many *different* languages from the *same* domain. Moreover, most of these methods need to pretrain language models or NMT models with a reasonable amount of data, and accuracy can suffer in low-resource settings like those encountered for LRLs (Duh et al., 2013).

In this chapter, we create a mathematical framework for data selection in multilingual MT that selects data from *all* languages, such that minimizing the training objective over the sampled data approximately minimizes the loss of the LRL MT model. The formulation leads to a simple, efficient, and effective algorithm that first samples a target sentence and then conditionally samples which of several source sentences to use for training. We name the method Target Conditioned Sampling (TCS). We also propose and experiment with several design choices for TCS, which are especially effective for LRLs. On the TED multilingual corpus (Qi et al., 2018), TCS leads to large improvements of up to 2 BLEU on three of the four languages we test, and no degradation on the fourth, with only slightly increased training time. To our knowledge, this is the first successful application of data selection to multilingual NMT.

3.2 Method

3.2.1 Multilingual Training Objective

First, in this section we introduce our problem formally, where we use the upper case letters X, Y to denote the random variables, and the corresponding lower case letters x, y to denote their actual values. Suppose our objective is to learn parameters θ of a translation model from a source language s into target language t. Let x be a source sentence from s, and y be the equivalent target sentence from t, given loss function $\mathcal{L}(x, y; \theta)$ our objective is to find optimal parameters θ^* that minimize:

$$\mathbb{E}_{x,y\sim P_S(X,Y)}[\mathcal{L}(x, y; \theta)] \tag{3.1}$$

where $P_s(X, Y)$ is the data distribution of s-t parallel sentences.

Unfortunately, we do not have enough data to accurately estimate θ^*, but instead we have a multilingual corpus of parallel data from languages $\{s_1, S_2, ..., S_n\}$ all into t. Therefore, we resort to multilingual training to facilitate the learning of θ. Formally, we want to construct a distribution $Q(X, Y)$ with support over $s_1, s_2, ..., s_n$-T to augment the s-t data with samples from Q during training. Intuitively, a good $Q(X, Y)$ will have an expected loss

$$\mathbb{E}_{x,y\sim Q(X,Y)}[\mathcal{L}(x, y; \theta)] \tag{3.2}$$

that is correlated with Eq. 3.1 over the space of all θ, so that training over data sampled from $Q(X, Y)$ can facilitate the learning of θ. Next, we explain a version of $Q(X, Y)$ designed to promote efficient multilingual training.

3.2.2 Target Conditioned Sampling

We argue that the optimal $Q(X, Y)$ should satisfy the following two properties.

First, $Q(X, Y)$ and $P_s(X, Y)$ should be **target invariant**; the marginalized distributions $Q(Y)$ and $P_s(Y)$ should match as closely as possible:

$$Q(Y) \approx P_s(Y) \tag{3.3}$$

This property ensures that Eq. 3.1 and Eq. 3.2 are optimizing towards the same target Y distribution.

Second, to have Eq. 3.2 correlated with Eq. 3.1 over the space of all θ, we need $Q(X, Y)$ to be correlated with $P_s(X, Y)$, which can be loosely written as

$$Q(X, Y) \approx P_s(X, Y). \tag{3.4}$$

Because we also make the target invariance assumption in Eq. 3.3,

$$\frac{Q(X, Y)}{Q(Y)} \approx \frac{P_s(X, Y)}{P_s(Y)} \tag{3.5}$$

$$Q(X|Y) \approx P_s(X|Y). \tag{3.6}$$

We call this approximation of $P_s(X|Y)$ by $Q(X|Y)$ **conditional source invariance**. Based on these two assumptions, we define Target Conditioned Sampling (TCS), a training framework that first samples $y \sim Q(Y)$, and then conditionally samples $x \sim Q(X|y)$ during training. Note $P_s(X|Y = y)$ is the optimal back-translation distribution, which implies that back-translation (Sennrich et al., 2016a) is a particular instance of TCS.

Of course, we do not have enough s-t parallel data to obtain a good estimate of the *true* back-translation distribution $P_s(X|y)$ (otherwise, we can simply use that data to learn θ). However, we posit that even a small amount of data is sufficient to construct an adequate data selection policy $Q(X|y)$ to sample the sentences x from multilingual data for training. Thus, the training objective that we optimize is

$$\mathbb{E}_{y\sim Q(Y)}\mathbb{E}_{x\sim Q(X|y)}\left[\mathcal{L}(x, y; \theta)\right] \tag{3.7}$$

Next, in §3.2.3, we discuss the choices of $Q(Y)$ and $Q(X|y)$.

3.2.3 Choosing the Sampling Distributions

Choosing $Q(Y)$. Target invariance requires that we need $Q(Y)$ to match $P_s(Y)$, which is the distribution over the target of s-t. We have parallel data from multiple languages $s_1, s_2, ..., s_n$, all into t. Assuming no systematic inter-language distribution differences, a uniform sample of a target sentence y from the multilingual data can approximate $P_s(Y)$. We thus only need to sample y uniformly from the union of all extra data.

Choosing $Q(X|y)$. Choosing $Q(X|y)$ to approximate $P_s(X|y)$ is more difficult, and there are a number of methods could be used to do so. To do so, we note that conditioning on the same target y and restricting the support of $P_s(X|y)$ to the sentences that translate into y in at least one of s_i-t, $P_s(X = x|y)$ simply measures how likely x is in s. We thus define a heuristic function $\text{sim}(x, s)$ that approximates the probability that x is a sentence in s, and follow the data augmentation objective in Wang et al. (2018a) in defining this probability according to

$$Q^*(x|y) = \frac{\exp\left(\text{sim}(x, s)/\tau\right)}{\sum_{x'} \exp\left(\text{sim}(x', s)/\tau\right)} \tag{3.8}$$

where is a temperature parameter that adjusts the peakiness of the distribution.

3.2.4 Algorithms

The formulation of $Q(X, Y)$ allows one to sample multilingual data with the following algorithm:

1. Select the target y based on $Q(y)$. In our case we can simply use the uniform distribution.
2. Given the target y, gather all data $(x_i, y) \in s_1, s_2, ...s_n$-$t$ and calculate $\text{sim}(x_i, s)$
3. Sample (x_i, y) based on $Q(X|y)$

The algorithm requires calculating $Q(X|y)$ repeatedly during training. To reduce this overhead, we propose two strategies for implementation: 1) **Stochastic**: compute $Q(X|y)$ before training starts, and dynamically sample each minibatch using the precomputed $Q(X|y)$; 2) **Deterministic**: compute $Q(X|y)$ before training starts and select $x' = \text{argmax}_x Q(x|y)$ for training. The deterministic method is equivalent to setting τ, the degree of diversity in $Q(X|y)$, to be 0.

3.2.5 Similarity Measure

In this section, we define two formulations of the similarity measure $\text{sim}(s, x)$, which is essential for constructing $Q(X|y)$. Each of the similarity measures can be calculated at two granularities: 1) language level, which means we calculate one similarity score for each language based on all of its training data; 2) sentence level, which means we calculate a similarity score for each sentence in the training data.

Vocab Overlap provides a crude measure of surface form similarity between two languages. It is efficient to calculate, and is often quite effective, especially for low-resource languages. Here we use the number of character n-grams that two languages share to measure the similarity between the two languages.

We can calculate the language-level similarity between S_i and S

$$\text{sim}_{\text{vocab-lang}}(s_i, s) = \frac{|\text{vocab}_k(s) \cap \text{vocab}_k(s_i)|}{k}$$

$\text{vocab}_k(\cdot)$ represents the top k most frequent character n-grams in the training data of a language. Then we can assign the same language-level similarity to all the sentences in s_i.

This can be easily extended to the sentence level by replacing $\text{vocab}_k(s_i)$ to the set of character n-grams of all the words in the sentence x.

Language Model trained on s can be used to calculate the probability that a data sequence belongs to s. Although it might not perform well if s does not have enough training data, it may still be sufficient for use in the TCS algorithm. The language-level metric is defined as

$$\text{sim}_{\text{LM-lang}}(s_i, s) = \exp\left(\frac{\sum_{c_i \in s_i} \text{NLL}_s(c_i)}{|c_i \in s_i|}\right)$$

where $\text{NLL}_s(\cdot)$ is negative log likelihood of a character-level LM trained on data from s. Similarly, the corresponding sentence level metric is the LM probability over each sentence x.

3.3 Experiment

3.3.1 Dataset and Baselines

We use the 58-language-to-English TED dataset (Qi et al., 2018). Following the setup in prior work (Qi et al., 2018; Neubig and Hu, 2018), we use three low-resource languages Azerbaijani (aze), Belarusian (bel), Galician (glg) to English, and a slightly higher-resource dataset, Slovak (slk) to English.

We use multiple settings for baselines: 1) Bi: each LRL is paired with its related HRL, following Neubig and Hu (2018). The statistics of the LRL and their corresponding HRL are listed in Table 6.2; 2) All: we train a model on all 58 languages; 3) Copied: following Currey et al. (2017), we use the union of all English sentences as monolingual data by copying them to the source side.

3.3.2 Experiment Settings

A standard sequence-to-sequence (Sutskever et al., 2014) NMT model with attention is used for all experiments. Byte Pair Encoding (BPE) (Sennrich et al., 2016b; Kudo and Richardson, 2018) with vocabulary size of 8000 is applied for each language individually.

3.3.3 Results

We test both the Deterministic (TCS-D) and Stochastic (TCS-S) algorithms described in §3.2.4. For each algorithm, we experiment with the similarity measures introduced in §3.2.5. The results are listed in Tab. 6.3.

Of all the baselines, Bi in general has the best performance, while All, which uses all the data and takes much longer to train, generally hurts the performance. This is consistent with findings in prior work (Neubig and Hu, 2018). Copied is only competitive for slk, which indicates the gain of TCS is not simply due to extra English data.

TCS-S combined with the language-level similarity achieves the best performance for all four languages, improving around 1 BLEU over the best baseline for aze, and around 2 BLEU for glg and slk. For bel, TCS leads to no degradation while taking much less training time than the best baseline All.

Sim	Method	aze	bel	glg	slk
-	Bi	10.35	15.82	27.63	26.38
-	All	10.21	17.46	26.01	26.64
-	copied	9.54	13.88	26.24	26.77
Back-Translate	TCS	7.79	11.50	27.45	28.44
LM-sent	TCS-D	10.34	14.68	27.90	27.29
LM-sent	TCS-S	10.95†	17.15	27.91	27.24
LM-lang	TCS-D	10.76	14.97	27.92	28.40
LM-lang	TCS-S	**11.47***	17.61	28.53†	**28.56***
Vocab-sent	TCS-D	10.68	16.13	27.29	27.03
Vocab-sent	TCS-S	11.09†	16.30	28.36†	27.01
Vocab-lang	TCS-D	10.58	16.32	28.17	28.27*
Vocab-lang	TCS-S	**11.46***	**17.79**	**29.57***	28.45*

Table 3.1: BLEU scores on four languages. Statistical significance Clark et al. (2011a) is indicated with $*$ ($p < 0.001$) and † ($p < 0.05$), compared with the best baseline.

TCS-D vs. TCS-S. Both algorithms, when using document-level similarity, improve over the baseline for all languages. TCS-D is quite effective without any extra sampling overhead. TCS-S outperforms TCS-D for all experiments, indicating the importance of diversity in the training data.

Sent. vs. Lang. For all experiments, language-level outperforms the sentence-level similarity. This is probably because language-level metric provides a less noisy estimation, making $Q(x|y)$ closer to $P_s(x|y)$.

LM vs. Vocab. In general, the best performing methods using LM and Vocab are comparable, except for glg, where Vocab-lang outperforms LM-lang by 1 BLEU. Slk is the only language where LM outperformed Vocab in all settings, probably because it has the largest amount of data to obtain a good language model. These results show that easy-to-compute language similarity features are quite effective for data selection in low-resource languages.

Back-Translation TCS constructs $Q(X|y)$ to sample augmented multilingual data, when the LRL data cannot estimate a good back-translation model. Here we confirm this intuition by replacing the $Q(X|y)$ in TCS with the back-translations generated by the model trained on the LRLs. To make it comparable to Bi, we use the sentence from the LRL and its most related HRL if there is one for the sampled y, but use the back-translated sentence otherwise. Tab. 6.3 shows that for slk, back-translate

achieves comparable results with the best similarity measure, mainly because slk has enough data to get a reasonable back-translation model. However, it performs much worse for aze and bel, which have the smallest amount of data.

3.4 Conclusion

We propose Target Conditioned Sampling (TCS), an efficient data selection framework for multilingual data by constructing a data sampling distribution that facilitates the NMT training of LRLs. TCS brings up to 2 BLEU improvements over strong baselines with only slight increase in training time.

Chapter 4

Optimizing Data Usage via Differentiable Rewards

Our method in the previous chapter shows that well-designed heuristic data selection methods can lead to significant improvements to multilingual training. This insight leads to a natural follow-up question: can we design an algorithm that automatically *learns* what data to use? To acquire a new skill, humans learn better and faster if a tutor informs them of how much attention they should pay to particular content or practice problems based on their current knowledge level. Similarly, a machine learning model could potentially be trained better if data is presented in a way that adapts to its current learning state. In this chapter, we examine the problem of training an adaptive scorer that weights data instances to maximally benefit learning.

4.1 Introduction

While deep learning models are remarkably good at fitting large data sets, their performance is also highly sensitive to the structure and domain of their training data. Training on out-of-domain data can lead to worse model performance, while using more relevant data can assist transfer learning. Previous work has attempted to create strategies to handle this sensitivity by selecting subsets of the data to train the model on (Jiang and Zhai, 2007; Wang et al.; Axelrod et al., 2011; Moore and Lewis, 2010), providing different weights for each example (Sivasankaran et al., 2017; Ren et al., 2018), or changing the presentation order of data (Bengio et al., 2009; Kumar et al., 2019).

However, there are several challenges with existing work on better strategies for data usage. Most data filtering critera or training curriculum rely on domain-specific knowledge and hand-designed heuristics, which can be sub-optimal. To avoid hand-designed heuristics, several works propose to optimize a parameterized neural network to learn the data usage schedule, but most of them are tailored to specific use cases, such as handling noisy data for classification (Jiang et al., 2018), learning a curriculum learning strategy for specific tasks (Kumar et al., 2019; Tsvetkov et al., 2016), and actively selecting data for annotation (Fang et al., 2017; Wu et al., 2018). Fan et al. (2018b) proposes a more general teacher-student

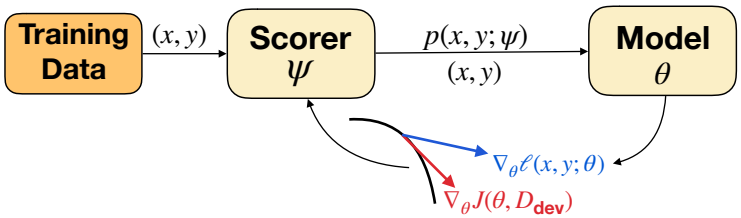

Figure 4.1: The general workflow of DDS.

framework that first trains a "teacher network" to select data that directly optimizes development set accuracy over multiple training runs. However, because running multiple runs of training simply to learn this teacher network entails an n-fold increase in training time for n runs, this is infeasible in many practical settings. In addition, in preliminary experiments we also found the single reward signal provided by dev set accuracy at the end of training to be noisy, to the extent that we were not able to achieve results competitive with simpler heuristic training methods.

In this chapter, we propose an alternative: a general Reinforcement Learning (RL) framework for optimizing training data usage by training a *scorer network* that minimizes the model loss on the development set. We formulate the scorer network as a function of the current training examples, and update the scorer along with the main model being trained. Thus, our method requires no heuristics and is generalizable to various tasks. To make the scorer adaptive, we train it using frequent and efficient updates towards a reward function inspired by recent work on learning from auxiliary tasks (Du et al., 2018; Liu et al., 2019b), which uses the similarity between two gradients as a measure of task relevance. We propose to use the gradient alignment between the training examples and the dev set as a reward signal for *a parametric scorer network*, as illustrated in Fig. 9.3. We then formulate our framework as a bi-level optimization problem similar to those found in prior works such as meta-learning for few-shot learning (Finn et al., 2017), noisy data filtering (Ren et al., 2018), and neural architecture search (Liu et al., 2019a), and demonstrate that our proposed update rules follow a direct differentiation of the scorer parameters to optimize the model loss on the dev set. Thus we refer to our framework as "Differentiable Data Selection" (DDS).

We demonstrate two concrete instantiations of the DDS framework, one for a more general case of image classification, and the other for a more specific case of NMT. For image classification, we test on both CIFAR-10 and ImageNet. For NMT, we focus on a multilingual setting, where we optimize data usage from a multilingual corpus to improve the performance on a particular language. For these two very different and realistic tasks, we find the DDS framework brings significant improvements over diverse baselines for all settings.

4.2 Differentiable Data Selection

4.2.1 Risk, Training, and Development Sets

Commonly in machine learning, we seek to find the parameters θ^* that minimize the *risk* $J(\theta, P)$, the expected value of a loss function $\ell(x, y; \theta)$, where $\langle x, y \rangle$ are pairs of inputs and associated labels sampled from a particular distribution $P(X, Y)$:

$$\begin{aligned} \theta^* &= \operatorname*{argmin}_{\theta} J(\theta, P) \quad \text{where} \\ J(\theta, P) &= \mathbb{E}_{x,y \sim P(X,Y)}[\ell(x, y; \theta)] \end{aligned} \tag{4.1}$$

Ideally, we would like the risk $J(\cdot)$ to be minimized over the data distribution that our system sees at test time, ie. $P_{\text{test}}(X, Y)$. Unfortunately, this distribution is unknown at training time, so instead we collect a training set $\mathcal{D}_{\text{train}} = \{(x_i, y_i) : i = 1, ..., N_{\text{train}}\}$ with distribution $P_{\text{train}}(X, Y) = \text{Uniform}(\mathcal{D}_{\text{train}})$, and minimize the *empirical risk* by taking $\langle x, y \rangle \sim P_{\text{train}}(X, Y)$. Since we need a sufficiently large training set $\mathcal{D}_{\text{train}}$ to train a good model, it is hard to ensure that $P_{\text{train}}(X, Y) \approx P_{\text{test}}(X, Y)$. In fact, we often accept that training data comes from a different distribution than test data. The discrepancy between $P_{\text{train}}(X, Y)$ and $P_{\text{test}}(X, Y)$ manifests itself in the form of problems such as overfitting (Zhang et al., 2017; Srivastava et al., 2014), covariate shift (Shimodaira, 2000), and label shift (Lipton et al., 2018).

However, unlike the large training set, we can often collect a relatively small development set $\mathcal{D}_{\text{dev}} = \{(x_i, y_i) : i = 1, ..., N_{\text{dev}}\}$ with a distribution $P_{\text{dev}}(X, Y)$ that is much closer to $P_{\text{test}}(X, Y)$ (Some examples can be found in §4.4). Since $\mathcal{D}_{\text{dev}}$ is a better approximation of our test-time scenario, we can use $\mathcal{D}_{\text{dev}}$ to get reliable feedback to learn to better utilize our training data from $\mathcal{D}_{\text{train}}$. In particular, we propose to train a *scorer* network, parameterized by ψ, that adjusts the weights of examples in $\mathcal{D}_{\text{train}}$ to minimize $J(\theta, \mathcal{D}_{\text{dev}})$.

4.2.2 Learning to Optimize Data Usage

We propose to optimize the scorer's parameters ψ in an RL setting. Our *environment* is the model state θ and an example $\langle x, y \rangle$. Our RL *agent* is the scorer network ψ, which optimizes the data usage for the current model state. The agent's *reward* on picking an example approximates the dev set performance of the resulting model after the model is updated on this example.

Our scorer network is parameterized as a differentiable function that only takes as inputs the features of the example $\langle x, y \rangle$. Intuitively, it represents a distribution over the training data where more important data has a higher probability of being used, denoted $P(X, Y; \psi)$. Unlike prior methods which generally require complicated featurization of both the model state and the data as input to the RL agent (Fan et al., 2018b; Jiang et al., 2018; Fang et al., 2017), our formulation is much simpler and generalizable to different tasks. Since our scorer network does not consider the model parameters θ_t as input, we update it iteratively with the model so that at training step t, $P(X, Y; \psi_t)$ provides an up-to-date data scoring feedback for a given θ_t.

Although the above formulation is simpler and more general, it requires much more frequent updates

to the scorer parameter ψ. Existing RL frameworks simply use the change in dev set risk as the regular reward signal, which makes the update expensive and unstable (Fan et al., 2018b; Kumar et al., 2019). Therefore, we propose a novel reward function as an approximation to $\Delta J_{\text{dev}}(x, y)$ to quantify the effect of the training example $\langle x, y \rangle$. Inspired by Du et al. (2018), which uses gradient similarity between two tasks to measure the effect of adaptating between them, we use the agreement between the model gradient on data $\langle x, y \rangle$ and the gradient on the dev set to approximate the effect of $\langle x, y \rangle$ on dev set performance. This reward implies that we prefer data that moves θ in the direction that minimizes the dev set risk:

$$R(x, y) = \Delta J_{\text{dev}}(x, y) \approx \nabla_\theta \ell(x, y; \theta_{t-1})^\top \cdot \nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}}) \tag{4.2}$$

According to the REINFORCE algorithm (Williams, 1992), the update rule for ψ is thus

$$\psi_{t+1} \leftarrow \psi_t + \underbrace{\nabla_\theta \ell(x, y; \theta_{t-1}) \cdot \nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})}_{R(x,y)} \nabla_\psi \log(P(X, Y; \psi)) \tag{4.3}$$

The update rule for the model is simply

$$\theta_t \leftarrow \theta_{t-1} - \nabla_\theta J(\theta_{t-1}, P(X, Y; \psi)) \tag{4.4}$$

For simplicity of notation, we omit the learning rate term. By alternating between Eq. 4.4 and Eq. 4.3, we can iteratively update θ using the guidance from the scorer network, and update ψ to optimize the scorer using feedback from the model.

Our formulation of scorer network as $P(X, Y; \psi)$ has several advantages. First, it provides the flexibility that we can either (1) sample a training instance with probability proportional to its score, (2) or equivalently scale the update from the training instance based on its score. In later sections, we provide an algorithm under the DDS framework for multilingual NMT (see §4.3.2), where the former is more efficient, and another more general algorithm for image classification (see §4.3.1), where the latter choice is natural. Second, it allows easy integration of prior knowledge of the data, which is shown to be effective in §4.4.

4.2.3 Deriving Rewards through Direct Differentiation

In this section, we show that the update for the scorer network in Eq. 4.3 can be approximately derived as the solution of a bi-level optimization problem (Colson et al., 2007), which has been applied to many different lines of research in the field of meta-learning (Baydin et al., 2018; Liu et al., 2019a; Ren et al., 2018).

Under our framework, the scorer samples the data according to $\langle x, y \rangle \sim P(X, Y; \psi)$, and ψ will be chosen so that θ^* that minimizes $J(\theta, P(X, Y; \psi))$ will approximately minimize $J(\theta, P_{\text{dev}}(X, Y))$:

$$\psi^* = \operatorname*{argmin}_\psi J(\theta^*(\psi), \mathcal{D}_{\text{dev}}) \text{ where } \theta^*(\psi) = \operatorname*{argmin}_\theta \mathbb{E}_{x,y \sim P(X,Y;\psi)} \left[\ell(x, y; \theta) \right] \tag{4.5}$$

The connection between ψ and θ in Eq. 4.5 shows that $J(\theta_t, \mathcal{D}_{\text{dev}})$ is differentiable with respect to ψ. Now we can approximately compute the gradient $\nabla_\psi J(\theta_t, \mathcal{D}_{\text{dev}})$ as follows:

$$
\begin{aligned}
\nabla_\psi J(\theta_t, \mathcal{D}_{\text{dev}}) &= \nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \theta_t(\psi) && \text{(apply chain rule:)} \\
&= \nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \left(\theta_{t-1} - \nabla_\theta J(\theta_{t-1}, \psi)\right) && \text{(substitute } \theta_t \text{ from Eq. 4.4:)} \\
&\approx -\nabla_{\theta_t} J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\psi \left(\nabla_\theta J(\theta_{t-1}, \psi)\right) && \text{(assume } \nabla_\psi \theta_{t-1} \approx 0\text{:)} \\
&= -\nabla_\psi \mathbb{E}_{x,y \sim P(X,Y;\psi)} \left[\nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1})\right] \\
&= -\mathbb{E}_{x,y \sim P(X,Y;\psi)} \left[\left(\nabla_\theta J(\theta_t, \mathcal{D}_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1})\right) \cdot \nabla_\psi \log P(x, y; \psi)\right]
\end{aligned}
\tag{4.6}
$$

Here, we make a Markov assumption that $\nabla_\psi \theta_{t-1} \approx 0$, assuming that at step t, given θ_{t-1} we do not care about how the values of ψ from previous steps led to θ_{t-1}. Intuitively, this assumption indicates in the previous step ψ_{t-1} is already updated regards to θ_{t-1}, so the effect of ψ on θ_{t-1} is likely to be minimal. This assumption can simplify and speed up computation. Moreover, this allows us to have a natural interpretation of the update rule for the data scorer: it should up-weight the training data that have similar gradient direction with the dev data[1]. Eq. 4.6 leads to a rule to update ψ using gradient descent, which is exactly the same as the RL update rule in Eq. 4.3.

4.2.4 Additional Clarifications

One potential concern with our approach is that because we optimize ψ_t directly on the dev set using $J(\theta_t, \mathcal{D}_{\text{dev}})$, we may risk indirectly overfitting model parameters θ_t by selecting a small subset of data that is overly specialized. However we do not observe this problem in practice, and posit that this because (1) the influence of ψ_t on the final model parameters θ_t is quite indirect, and acts as a "bottleneck" which has similarly proven useful for preventing overfitting in neural models (Grézl et al., 2007), and (2) because the actual implementations of DDS (which we further discuss in §4.3) only samples a subset of data from $\mathcal{D}_{\text{train}}$ at each optimization step, further limiting expressivity.

4.3 Concrete Instantiations of DDS

We now turn to discuss two concrete instantiations of DDS that we use in our experiments: a more generic example of classification, which should be applicable to a wide variety of tasks, and a specialized application to the task of multilingual NMT, which should serve as an example of how DDS can be adapted to the needs of specific applications.

4.3.1 Formulation for Classification

Alg. 1 presents the pseudo code for the training process on classification tasks, using the notation introduced in §5.2.

[1] Our use of the Markov assumption is based on its use and empirical success in previous work on bi-level optimization, such as Hyper Gradient Descent (Baydin et al. 2017) and many others. Of course, this is a simplifying assumption, but we believe that our empirical results show that the proposed method is useful nonetheless.

Algorithm 1: Training a classification model with DDS.

Input : $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{dev}}$

Output: Optimal parameters θ^*

Initializer θ_0 and ψ_0

for $t = 1$ **to** *num_train_steps* **do**

  Sample B training data points $x_i, y_i \sim \text{Uniform}(\mathcal{D}_{\text{train}})$

  Sample B validation data points $x'_i, y'_i \sim \text{Uniform}(\mathcal{D}_{\text{dev}})$

  ▷ *Optimize* θ

  $g_\theta \leftarrow \sum_{i=1}^{B} p(x_i, y_i; \psi_{t-1}) \nabla_\theta \ell(x_i, y_i; \theta_{t-1})$

  Update $\theta_t \leftarrow \text{GradientUpdate}\Big(\theta_{t-1}, g_\theta\Big)$

  ▷ *Evaluate* θ_t *on* $\mathcal{D}_{dev}$

  Let $d_\theta \leftarrow \frac{1}{B} \sum_{j=1}^{B} \nabla_\theta \ell(x'_j, y'_j; \theta_t)$

  ▷ *Optimize* ψ

  $r_i \leftarrow d_\theta^\top \cdot \nabla_\theta \ell(x_i, y_i; \theta_{t-1})$

  Let $d_\psi \leftarrow \frac{1}{B} \sum_{i=1}^{B} [r_i \cdot \nabla_\psi \log p(x_i, y_i; \psi)]$

  Update $\psi_t \leftarrow \text{GradientUpdate}(\psi_{t-1}, d_\psi)$

end

The main classification model is parameterized by θ. The scorer $p(X, Y; \psi)$ uses an architecture identical to the main model, but with independent weights, *i.e.* $p(X, Y; \psi)$ does not share weights with θ. For each example x_i in a minibatch uniformly sampled from $\mathcal{D}_{\text{train}}$[2], this DDS model outputs a scalar for the data point x_i. All scalars are passed through a softmax function to compute the relative probabilities of the examples in the minibatch, and their gradients are scaled accordingly when applied to θ.

We have two gradient update steps, one for the model parameter θ_t in line 6 and the other for the DDS scorer parameter ψ in line 10. For the model parameter update, we can simply use any of the standard optimization update rule. For the scorer ψ, we use the update rule derived in § 4.2.3.

Per-Example Gradient. In standard neural network training, a single aggregate gradient is computed with respect to a mini-batch of training data of size n to improve computational efficiency. In contrast, as seen from line 9 of Alg. 1, as well as from **??**, DDS requires us to compute $\nabla_\theta \ell(x_i, y_i; \theta_{t-1})$, the gradient for each example in a batch of training data. This potentially slows down training by a factor of n. A naive implementation of this operation would be very slow and memory intensive, especially when the batch size is large, *e.g.* our experiments on ImageNet use a batch size of 4096 (see §4.4).

We propose an efficient approximation of this per-example gradient computation via the first-order Taylor expansion of $\ell(x_i, y_i; \theta_{t-1})$. In particular, for any vector $v \in \mathbb{R}^{|\theta|}$, with sufficiently small $\epsilon > 0$, we have:

[2]Note that our actual formulation of $p(X, Y; \psi)$ does *not* depend on Y, but we keep Y in the notation for consistency with the formulation of the DDS framework.

$$
\begin{aligned}
& v^\top \cdot \nabla_\theta \ell(x_i, y_i; \theta_{t-1}) \\
& \approx \frac{1}{\epsilon} \left(\ell\big(x_i, y_i; \theta_{t-1} + \epsilon v\big) - \ell\big(x_i, y_i; \theta_{t-1}\big) \right),
\end{aligned}
\tag{4.7}
$$

Eq. 4.7 can be implemented by keeping a shadow version of parameters θ_{t-1}, caching training loss $\ell(x_i, y_i; \theta_{t-1})$, and computing the new loss with $\theta_{t-1} + \epsilon v$. Here, v is d_θ as in line 9 of Alg. 1.

4.3.2 Formulation for Multilingual NMT

Next we demonstrate an application of DDS to multilingual models for NMT, specifically for improving accuracy on low-resource languages (LRL) (Zoph et al., 2016; Neubig and Hu, 2018).

Problem Setting. In this setting, we assume that we have a particular LRL S that we would like to translate into target language T, and we additionally have a multilingual corpus $\mathcal{D}_{\text{train}}$ that has parallel data between n source languages $(S_1, S_2, ..., S_n)$ and target language T. We would like to pick parallel data from any of the source languages to the target language to improve translation of a particular LRL S, so we assume that $\mathcal{D}_{\text{dev}}$ exclusively consists of parallel data between S and T. Thus, DDS will select data from $\mathcal{D}_{\text{train}}$ that improve accuracy on S-to-T translation as represented by $\mathcal{D}_{\text{dev}}$.

Adaptation to NMT. To make training more efficient and stable in this setting, we make three simplifications of the main framework in §4.2.3 that take advantage of the problem structure of multilingual NMT. First, instead of directly modeling $p(X, Y; \psi)$, we assume a uniform distribution over the target sentence Y, and only parameterize the conditional distribution of which source language sentence to pick given the target sentence: $p(X|y; \psi)$. This design follows the formulation of Target Conditioned Sampling (TCS; **?**), an existing state-of-the-art data selection method that uses a similar setting but models the distribution $p(X|y)$ using heuristics. Since the scorer only needs to model a simple distribution over training languages, we use a fully connected 2-layer perceptron network, and the input is a vector indicating which source languages are available for the given target sentence. Second, we only update ψ after updating the NMT model for a fixed number of steps. Third, we sample the data according to $p(X|y; \psi)$ to get a Monte Carlo estimate of the objective in Eq. 4.5. This significantly reduces the training time compared to using all data. The pseudo code of the training process is in Alg. 3. In practice, we use cosine distance instead of dot product to measure the gradient alignment between the training and dev language, because cosine distance has smaller variance and thus makes the scorer update more stable.

4.4 Experiments

We now discuss experimental results on both image classification, an instance of the general classification problem using Alg. 1, and multilingual NMT using Alg. 3.

Algorithm 2: Training multilingual NMT with DDS.

Input : $\mathcal{D}_{\text{train}}$; K: number of data to train the NMT model before updating ψ; E: number of updates for ψ; α_1,α_2: discount factors for the gradient

Output: The converged NMT model θ^*

Initialize ψ_0, θ_0

▷ *Initialize the gradient of each source language*

$grad[S_i] \leftarrow 0$ **for** *i in n*

while θ *not converged* **do**

  $X, Y \leftarrow$ load_data$(\psi, \mathcal{D}_{\text{train}}, K)$

  ▷ *Train the NMT model*

  for x_i, y *in* X, Y **do**

    $\theta_t \leftarrow \text{GradientUpdate}\left(\theta_{t-1}, \nabla_{\theta_{t-1}} \ell(x_i, y; \theta_{t-1})\right)$

    $grad[S_i] \leftarrow \alpha_1 \times \text{grad}[S_i] + \alpha_2 \times \nabla_{\theta_{t-1}} \ell(x_i, y; \theta_{t-1})$

  end

  ▷ *Optimize* ψ

  for *iter in E* **do**

    sample B data pairs from $\mathcal{D}_{\text{train}}$

    $d_\psi \leftarrow \frac{1}{B} \sum_{j=1}^{B} \sum_{i=1}^{n} \left[\text{grad}[S_i]^\top \text{grad}[S] \cdot \nabla_{\psi_{t-1}} \log\left(p\left(S_i | y_j; \psi_{t-1}\right)\right)\right]$

    $\psi_t \leftarrow \text{GradientUpdate}(\psi_{t-1}, d_{\psi_{t-1}})$

  end

end

Table 4.1: Results for image classification accuracy (left) and multilingual MT BLEU (right). For MT, the statistical significance is indicated with $*$ ($p < 0.005$) and † ($p < 0.0001$). DDS outperforms the best baseline in all settings. For both image classification and NMT, DDS performs better than other intelligent data selection methods.

Methods	CIFAR-10 (WRN-28-k)		ImageNet (ResNet-50)	
	4K, $k = 2$	Full, $k = 10$	10%	Full
Uniform	82.60±0.17	95.55±0.15	56.36/79.45	76.51/93.20
SPCL	81.09±0.22	93.66±0.12	-	-
BatchWeight	79.61±0.50	94.11±0.18	-	-
MentorNet	83.11±0.62	94.92±0.34	-	-
DDS	83.63± 0.29	96.31± 0.13	**56.81/79.51**	**77.23/93.57**
retrained DDS	**85.56±0.20**	**97.91±0.12**	-	-

Methods	**aze**	**bel**	**glg**	**slk**
Uniform	10.31	17.21	26.05	27.44
SPCL	9.07	16.99	23.64	21.44
Related	10.34	15.31	27.41	25.92
TCS	11.18	16.97	27.28	27.72
DDS	10.74	17.24	27.32	**28.20***
TCS+DDS	**11.84***	**17.74**†	**27.78**	27.74

4.4.1 Experimental Settings

Data. We apply our method on established benchmarks for image classification and multilingual NMT. For image classification, we use CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015). For each dataset, we consider two settings: a reduced setting where only roughly 10% of the training labels are used, and a full setting, where all labels are used. Specifically, the reduced setting for CIFAR-10 uses the first 4000 examples in the training set, and with ImageNet, the reduced setting uses the first 102 TFRecord shards as pre-processed by Kornblith et al. (2019). We use the size of 224×224 for ImageNet.

For multilingual NMT, we use the 58-language-to-English TED dataset (Qi et al., 2018). Following prior work (Qi et al., 2018; Neubig and Hu, 2018; Wang et al., 2019b), we evaluate translation from four low-resource languages (LRL) Azerbaijani (`aze`), Belarusian (`bel`), Galician (`glg`), and Slovak (`slk`) to English, where each is paired with a similar high-resource language Turkish (`tur`), Russian (`rus`), Portugese (`por`), and Czech (`ces`). We combine data from all 8 languages, and use DDS to optimize data selection for each LRL.

To update the scorer, we construct $\mathcal{D}_{\text{dev}}$ so that it does not overlap with $\mathcal{D}_{\text{test}}$. For image classification, we hold out 10% of the training data as $\mathcal{D}_{\text{dev}}$; while for multilingual NMT, we simply use the dev set of the LRL as $\mathcal{D}_{\text{dev}}$.

Models and Training Details. For image classification, on CIFAR-10, we use the pre-activation WideResNet-28 (Zagoruyko and Komodakis, 2016), with width factor $k = 2$ for the reduced setting and $k = 10$ for the normal setting. For ImageNet, we use the post-activation ResNet-50 (He et al., 2016).

For NMT, we use a standard LSTM-based attentional baseline (Bahdanau et al., 2015), which is similar to previous models used in low-resource scenarios on this dataset (Neubig and Hu, 2018; Wang et al., 2019b) and others (Sennrich and Zhang, 2019b) due to its relative stability compared to other options such as the Transformer (Vaswani et al., 2017). Accuracy is measured using BLEU score (Papineni et al., 2002).

Baselines and Our Methods. For both image classification and multi-lingual NMT, we compare the

following data selection methods. **Uniform**: data is selected uniformly from all of the data that we have available, as is standard in training models. **SPCL** (Jiang et al., 2015): a curriculum learning method that dynamically updates the curriculum to focus more on the "easy" training examples based on model loss. **DDS**: our proposed method.

For image classification, we compare with several additional methods designed for filtering noisy data on CIFAR-10, where we simply consider the dev set as the clean data. **BatchWeight** (Ren et al., 2018): a method that scales example training loss in a batch with a locally optimized weight vector using a small set of clean data. **MentorNet** (Jiang et al., 2018): a curriculum learning method that trains a mentor network to select clean data based on features from both the data and the main model.

For machine translation, we also compare with two state-of-the-art heuristic methods for multi-lingual data selection. **Related**: data is selected uniformly from the target LRL and a linguistically related HRL (Neubig and Hu, 2018). **TCS**: the heuristic data selection method of "target conditioned sampling" introduced in Chapter 3, which uniformly chooses target sentences, then picks which source sentence to use based on heuristics such as word overlap. Note that both of these methods take advantage of structural properties of the multi-lingual NMT problem, and do not generalize to other problems such as classification.

DDS with Prior Knowledge DDS is a flexible framework to incorporate prior knowledge about the data using the scorer network, which can be especially important when the data has certain structural properties such as language or domain. We test such a setting of DDS for both tasks.

For image classification, we use **retrained DDS**, where we first train a model and scorer network using the standard DDS till convergence. The trained scorer network can be considered as a good prior over the data, so we use it to train the final model from scratch again using DDS. For multilingual NMT, we experiment with **TCS+DDS**, where we initialize the parameters of DDS with the TCS heuristic, then continue training.

4.4.2 Main Results

The results of the baselines and our method are listed in Tab. 6.3. First, comparing the standard baseline strategy of "Uniform" and the proposed method of "DDS" we can see that in all 8 settings DDS improves over the uniform baseline. This is a strong indication of both the consistency of the improvements that DDS can provide, and the generality – it works well in two very different settings. Next, we find that DDS outperforms SPCL by a large margin for both of the tasks, especially for multilingual NMT. This is probably because SPCL weighs the data only by their easiness, while ignoring their relevance to the dev set, which is especially important in settings where the data in the training set can have very different properties such as the different languages in multilingual NMT.

DDS also brings improvements over the state-of-the-art intelligent data utilization methods. For image classification, DDS outperforms MentorNet and BatchWeight on CIFAR-10 in all settings. For NMT, in comparison to Related and TCS, vanilla DDS performs favorably with respect to these state-of-the-art data selection baselines, outperforming each in 3 out of the 4 settings (with exceptions of

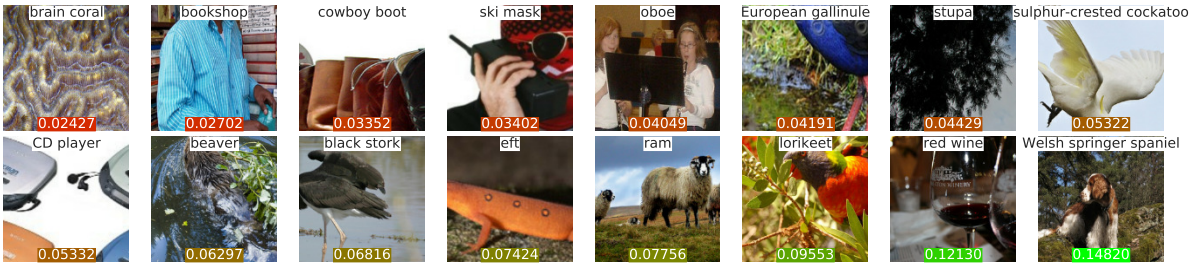

Figure 4.2: Example images from the ImageNet and their weights assigned by DDS. A trained DDS scorer assigns higher probabilities to images from ImageNet, in which the class content is more clear. Each image's label and weight in the minibatch is shown.

slightly underperforming Related on g1g and TCS on aze). In addition, we see that incorporating prior knowledge into the scorer network leads to further improvements. For image classification, retrained DDS can significantly improve over regular DDS, leading to the new state-of-the-art result on the CIFAR-10 dataset. For mulitlingual NMT, TCS+DDS achieves the best performance in three out of four cases (with the exception of s1k, where vanilla DDS already outperformed TCS).[3]

DDS does not incur much computational overhead. For image classification and multilingual NMT respectively, the training time is about $1.5\times$ and $2\times$ the regular training time without DDS. This contrasts favorably to previous methods that learn to select data using reinforcement learning. For example, in the IMDB movie review experiment in (Fan et al., 2018a), the data agent is trained for 200 episode, where each episode uses around 40% of the whole dataset, requiring 80x more training time than a single training run.

4.4.3 Analysis

Image classification. Prior work on heuristic data selection has found that models perform better when fed higher quality or more domain-relevant data towards the end of training (van der Wees et al., 2017b; Wang et al., 2019a). Here we verify this observation by analyzing the learned importance weight at the end of training for image classification. Fig. 4.3 shows that at the end of training, DDS learns to balance the class distribution, which is originally unbalanced due to the dataset creation. Fig. 4.2 shows that at the end of training, DDS assigns higher probabilities to images with clearer class content from ImageNet. These results show that DDS learns to focus on higher quality data towards the end of training.

NMT. Next, we focus on multilingual NMT, where the choice of data directly corresponds to picking a language, which has an intuitive interpretation. Since DDS adapts the data weights dynamically to the model throughout training, here we analyze how the dynamics of learned weights. Fig. 4.4 shows an

[3]Significance tests (Clark et al., 2011b) find significant gains over the baseline for aze, s1k, and bel. For g1g, DDS without heuristics performs as well as the TCS baseline.

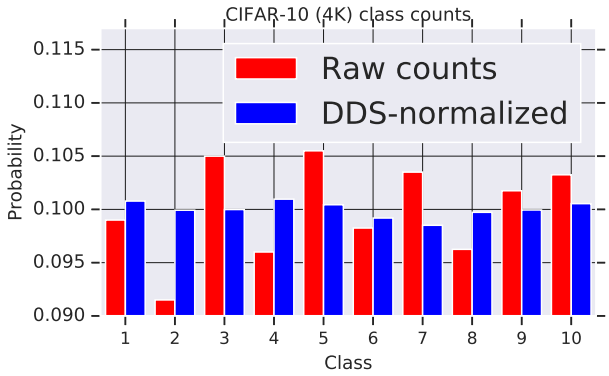

Figure 4.3: A trained DDS scorer learns to balance the class distributions of CIFAR-10 4K.

interesting trend of DDS without heuristic initialization. For both `aze` and `bel`, DDS focuses on the most related HRL after a certain number of training updates. Interestingly, for `bel`, DDS learns to focus on both `rus`, its most related HRL, and `ces`. Similarly for `slk`, DDS also learns to focus on `ces`, its most related HRL, and `rus`, although there is little vocabulary overlap between `slk` and `rus`. Also notably, the ratios change significantly over the course of training, indicating that different types of data may be more useful during different learning stages.

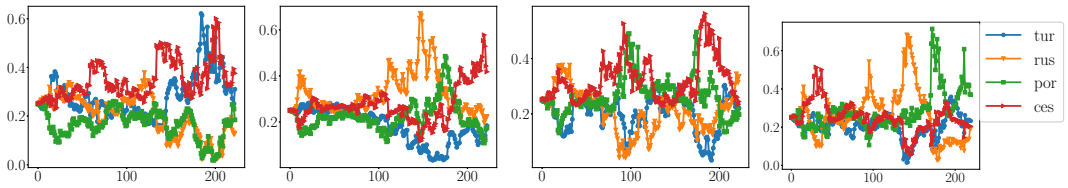

Figure 4.4: Language usage for DDS by training step. *From left to right*: `aze`, `bel`, `glg`, `slk`.

4.5 Related Work

Data Selection Methods Data selection for domain adaptation for disparate tasks has also been extensively studied (Moore and Lewis, 2010; Axelrod et al., 2011; Ngiam et al., 2019; Jiang and Zhai, 2007; Foster et al., 2010; Wang et al.). These methods generally design heuristics to measure domain similarity, while DDS is a more general data selection framework that works for both classification and other usage cases. Besides domain adaptation, data selection also benefits training in the face of noisy or otherwise undesirable data (Vyas et al., 2018; Pham et al., 2018). The idea of selecting training data that are similar to dev data has been used in works on submodular optimization (Kirchhoff and Bilmes, 2014; Tschiatschek et al., 2014), but they rely on features specific to the task, while DDS directly optimizes the the dev set performance, and is generalizable across tasks. Moreover, unlike DDS, these methods cannot adaptively change the data selection scheme.

Instance Weighting Methods Our method is also related to works on training instance weighting (Sivasankaran et al., 2017; Ren et al., 2018; Jiang and Zhai, 2007; Ngiam et al., 2019). These methods reweigh data based on a manually computed weight vector, instead of using a parameterized neural network. Notably, Ren et al. (2018) tackles noisy data filtering for image classification, by using meta-learning to calculate a locally optimized weight vector for each batch of data. In contrast, our work focuses on the general problem of optimizing data usage. We train a parameterized scorer network that optimizes over the entire data space, which can be essential in preventing overfitting mentioned in § 5.2; empirically our method outperform Ren et al. (2018) by a large margin in § 4.4. Sivasankaran et al. (2017) optimizes data weights by minimizing the error rate on the dev set. However, they use a single number to weigh each subgroup of augmented data, and their algorithm requires an expensive heuristic method to update data weights; while DDS uses a more expressive parameterized neural network to model the individual data weights, which are efficiently updated by directly differentiating the dev loss.

Curriculum Learning Many machine learning approaches consider how to best present data to models. First, difficulty-based curriculum learning estimates the presentation order based on heuristic understanding of the hardness of examples (Bengio et al., 2009; Spitkovsky et al., 2010; Tsvetkov et al., 2016; Zhang et al., 2016; Graves et al., 2017; Zhang et al., 2018; Platanios et al., 2019). These methods, though effective, often generalize poorly because they require task-specific difficulty measures. On the other hand, self-paced learning (**??**) defines the hardness of the data based on the loss from the model, but is still based on the assumption that the model should learn from easy examples. Our method does not make these assumptions.

RL for Training Data Usage Our method is closest to the learning to teach framework (Fan et al., 2018b) but their formulation involves manual feature design and requires expensive multi-pass optimization. Instead, we formulate our reward using bi-level optimization, which has been successfully applied for a variety of other tasks (Colson et al., 2007; Anandalingam and Friesz, 1992; Liu et al., 2019a; Baydin et al., 2018; Ren et al., 2018). (Wu et al., 2018; Kumar et al., 2019; Fang et al., 2017) propose RL frameworks for specific natural language processing tasks, but their methods are less generalizable and requires more complicated featurization.

4.6 Conclusion

We present *differentiable data selection*, an efficient RL framework for optimizing training data usage. We parameterize the scorer network as a differentiable function of the data, and provide an intuitive reward function for efficiently training the scorer network. We formulate two algorithms under the DDS framework for two realistic and very different tasks, image classification and multilingual NMT, which lead to consistent improvements over strong baselines.

Chapter 5

Balancing Training for Multilingual Neural Machine Translation

Equipped with the general automatic data selection method introduced in the previous chapter, we turn to a specific data balancing problem in multilingual training. As discussed in § 2.4, when training multilingual machine translation models that can translate to/from multiple languages, we are faced with imbalanced training sets: some languages have much more training data than others. In this chapter, we propose a method that automatically learns how to weight training data through a data scorer that is optimized to maximize performance on all test languages.[1]

5.1 Introduction

A common problem with multilingual training is that the data from different languages are both heterogeneous (different languages may exhibit very different properties) and imbalanced (there may be wildly varying amounts of training data for each language). Thus, while LRLs will often benefit from transfer from other languages, for languages where sufficient monolingual data exists, performance will often *decrease* due to interference from the heterogeneous nature of the data. This is especially the case for modestly-sized models that are conducive to efficient deployment (Arivazhagan et al., 2019; Conneau et al., 2019).

To balance the performance on different languages, the standard practice is to heuristically adjust the distribution of data used in training, specifically by over-sampling the training data from LRLs (Johnson et al., 2016b; Neubig and Hu, 2018; Arivazhagan et al., 2019; Conneau et al., 2019). For example, Arivazhagan et al. (2019) sample training data from different languages based on the dataset size scaled by a heuristically tuned temperature term. However, such heuristics are far from perfect. First, Arivazhagan et al. (2019) find that the exact value of this temperature term significantly affects results, and we further show in experiments that the ideal temperature varies significantly from one experimental setting to another. Second, this heuristic ignores factors other than data size that affect the interaction between

[1]The code is available at https://github.com/cindyxinyiwang/fairseq/tree/multiDDS.

different languages, despite the fact that language similarity has been empirically proven important in examinations of cross-lingual transfer learning (Wang and Neubig, 2019; Lin et al., 2019).

In this chapter, we ask the question: "is it possible to *learn* an optimal strategy to automatically balance the usage of data in multilingual model training?" To this effect, we propose a method that learns a language scorer that can be used throughout training to improve the model performance on *all* languages. Our method is based on the approach of Differentiable Data Selection (DDS) introduced in Chapter 4, a general machine learning method for optimizing the weighting of different training examples to improve a pre-determined objective. In this work, we take this objective to be the average loss from different languages, and directly optimize the weights of training data from each language to maximize this objective on a multilingual development set. This formulation has no heuristic temperatures, and enables the language scorer to consider the interaction between languages.

Based on this formulation, we propose an algorithm that improves the ability of DDS to optimize *multiple* model objectives, which we name MultiDDS. This is particularly useful in the case where we want to optimize performance on multiple languages simultaneously. Specifically, MultiDDS (1) has a more flexible scorer parameterization, (2) is memory efficient when training on multiple languages, and (3) stabilizes the reward signal so that it improves all objectives simultaneously instead of being overwhelmed by a single objective.

While the proposed methods are model-agnostic and thus potentially applicable to a wide variety of tasks, we specifically test them on the problem of training multilingual NMT systems that can translate many languages in a single model. We perform experiments on two sets of languages (one with more similarity between the languages, one with less) and two translation directions (one-to-many and many-to-one where the "one" is English). Results show that MultiDDS consistently outperforms various baselines in all settings. Moreover, we demonstrate MultiDDS provides a flexible framework that allows the user to define a variety of optimization objectives for multilingual models.

5.2 Multilingual Training Preliminaries

Monolingual Training Objective A standard NMT model is trained to translate from a single source language S to a target language T. The parameters of the model are generally trained by preparing a training dataset D_{train}, and defining the empirical distribution of sentence pairs $\langle x, y \rangle$ sampled from D_{train} as P. We then minimize the empirical risk $J(\theta, P)$, which is the expected value of the loss function $\ell(x, y; \theta)$ over this distribution:

$$\begin{aligned} \theta^* &= \operatorname*{argmin}_{\theta} J(\theta, D_{\text{train}}) \\ \text{where} \quad J(\theta, D_{\text{train}}) &= \mathbb{E}_{x,y \sim P(X,Y)}[\ell(x, y; \theta)] \end{aligned} \tag{5.1}$$

Multilingual Training Formulation A multilingual NMT model can translate n pairs of languages, from any source language S^i. to its corresponding target T^i. To train such a multilingual model, we have access to n sets of training data $D_{\text{train}} = D^1_{\text{train}}, D^2_{\text{train}}, \ldots, D^n_{\text{train}}$, where D^i_{train} is training data for language pair S^i-T^i. From these datasets, we can define P^i, the distribution of sentences from S^i-T^i,

and consequently also define a risk $J(\theta, P^i)$ for each language following the monolingual objective in Eq. 5.1.

However, the question now becomes: "how do we define an overall training objective given these multiple separate datasets?" Several different methods to do so have been proposed in the past. To discuss all of these different methods in a unified framework, we further define a distribution P_D over the n sets of training data, and define our overall multilingual training objective as

$$J_{\text{mult}}(\theta, P_D, D_{\text{train}}) = \mathbb{E}_{i\sim P_D(i;\psi)}\left[J(\theta, D^i_{\text{train}})\right]. \tag{5.2}$$

In practice, this overall objective can be approximated by selecting a language according to $\tilde{i} \sim P_D(i)$, then calculating gradients with respect to θ on a batch of data from $D^{\tilde{i}}_{\text{train}}$.

Evaluation Methods Another important question is how to evaluate the performance of such multilingual models. During training, it is common to use a separate development set for each language $D_{\text{dev}} = D^1_{\text{dev}}, D^2_{\text{dev}}, ..., D^n_{\text{dev}}$ to select the best model. Given that the objective of multilingual training is generally to optimize the performance on all languages simultaneously (Arivazhagan et al., 2019; Conneau et al., 2019), we can formalize this objective as minimizing the average of dev risks[2]:

$$J_{\text{dev}}(\theta, D_{\text{dev}}) = \frac{1}{n}\sum_{i=1}^{n} J(\theta, D^i_{\text{dev}}). \tag{5.3}$$

Relation to Heuristic Strategies This formulation generalizes a variety of existing techniques that define $P_D(i)$ using a heuristic strategy, and keep it fixed throughout training.

Uniform: The simplest strategy sets $P_D(i)$ to a uniform distribution, sampling minibatches from each language with equal frequency (Johnson et al., 2016b).

Proportional: It is also common to sample data in portions equivalent to the size of the corresponding corpora in each language (Johnson et al., 2016b; Neubig and Hu, 2018).

Temperature-based: Finally, because both of the strategies above are extreme (proportional under-weighting LRLs, and uniform causing overfitting by re-sampling sentences from limited-size LRL datasets), it is common to sample according to data size exponentiated by a temperature term τ (Arivazhagan et al., 2019; Conneau et al., 2019):

$$P_D(i) = \frac{q_i^{1/\tau}}{\sum_{k=1}^{n} q_k^{1/\tau}} \text{ where } q_i = \frac{|D^i_{\text{train}}|}{\sum_{k=1}^{n} |D^k_{\text{train}}|}. \tag{5.4}$$

When $\tau = 1$ or $\tau = \infty$ this is equivalent to proportional or uniform sampling respectively, and when a number in the middle is chosen it becomes possible to balance between the two strategies.

As noted in the introduction, these heuristic strategies have several drawbacks regarding sensitivity to the τ hyperparameter, and lack of consideration of similarity between the languages. In the following sections we will propose methods to resolve these issues.

[2] In reality, it is common to have the loss ℓ be a likelihood-based objective, but finally measure another metric such as BLEU score at test time, but for simplicity we will assume that these two metrics are correlated.

5.3 Differentiable Data Selection

Now we turn to the question: is there a better way to optimize $P_D(i)$ so that we can achieve our final objective of performing well on a representative development set over all languages, i.e. minimizing $J_{\text{dev}}(\theta, D_{\text{dev}})$. In order to do so, we turn to the method of Differentiable Data Selection (DDS, introduced in Chapter 4), a general purpose machine learning method that allows for weighting of training data to improve performance on a separate set of held-out data.

Specifically, DDS uses a technique called *bi-level optimization* (Colson et al., 2007), that learns a second set of parameters ψ that modify the training objective that we use to learn θ, so as to maximize the final objective $J_{\text{dev}}(\theta, D_{\text{dev}})$. Specifically, it proposes to learn a data scorer $P(x, y; \psi)$, parameterized by ψ, such that training using data sampled from the scorer optimizes the model performance on the dev set. To take the example of learning an NMT system to translate a *single* language pair i using DDS, the general objective in Eq. 5.1 could be rewritten as

$$\begin{aligned} \psi^* &= \underset{\psi}{\operatorname{argmin}}\, J(\theta^*(\psi), D^i_{\text{dev}}) \text{ where} \\ \theta^*(\psi) &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{x,y\sim P(x,y;\psi)} \left[\ell(x, y; \theta)\right]. \end{aligned} \tag{5.5}$$

DDS optimizes θ and ψ iteratively throughout the training process. Given a fixed ψ, the update rule for θ is simply

$$\theta_t \leftarrow \theta_{t-1} - \nabla_\theta \mathbb{E}_{x,y\sim P(x,y;\psi)} \left[\ell(x, y; \theta)\right]$$

To update the data scorer, DDS uses reinforcement learning with a reward function that approximates the effect of the training data on the model's dev performance

$$\begin{aligned} R(x, y; \theta_t) &\approx \nabla J(\theta_t, D^i_{\text{dev}})^\top \cdot \nabla_\theta \ell(x, y; \theta_{t-1}) \\ &\approx \cos\left(\nabla J(\theta_t, D^i_{\text{dev}}), \nabla_\theta \ell(x, y; \theta_{t-1})\right) \end{aligned} \tag{5.6}$$

where $\cos(\cdot)$ is the cosine similarity of two vectors. This reward can be derived by directly differentiating $J(\theta(\psi), D^i_{\text{dev}})$ with respect to ψ, but intuitively, it indicates that the data scorer should be updated to up-weigh the data points that have similar gradient with the dev data. According to the REINFORCE algorithm Williams (1992), the update rule for the data scorer then becomes

$$\psi_{t+1} \leftarrow \psi_t + R(x, y; \theta_t) \cdot \nabla_\psi \log P(x, y; \psi) \tag{5.7}$$

5.4 DDS for Multilingual Training

In this section, we use the previously described DDS method to derive a new framework that, instead of relying on fixed heuristics, adaptively optimizes usage of multilingual data for the best model performance on *multiple* languages. We illustrate the overall workflow in Fig. 5.1.

First, we note two desiderata for our multilingual training method: 1) **generality**: the method should be flexible enough so that it can be utilized universally for different multilingual tasks and settings (such as different translation directions for NMT). 2) **scalablity**: the method should be stable and efficient if one wishes to scale up the number of languages that a multilingual model supports. Based on these two properties, we introduce MultiDDS, an extension of the DDS method tailored for multilingual training.

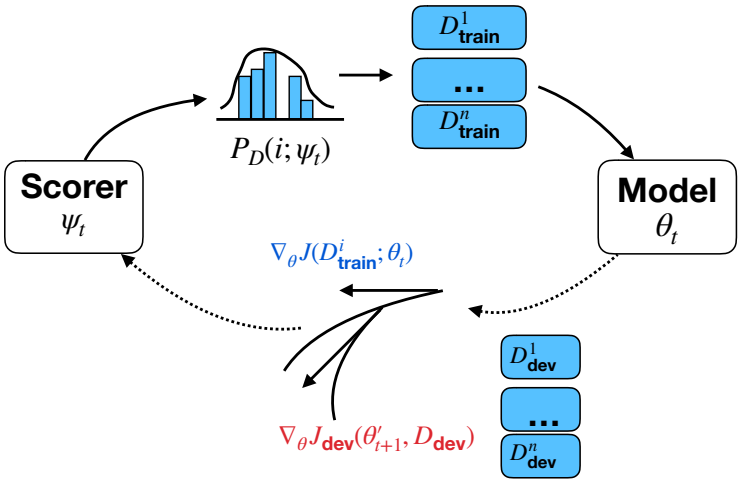

Figure 5.1: An illustration of the MultiDDS algorithm. Solid lines represent updates for θ, and dashed lines represent updates for ψ. The scorer defines the distribution over n training languages, from which training data is sampled to train the model. The scorer is updated to favor the datasets with similar gradients as the gradient of the aggregated dev sets.

Method MultiDDS directly parameterizes the standard dataset sampling distribution for multilingual training with ψ:

$$P_D(i;\psi) = e^{\psi_i} / \textstyle\sum_{k=1}^{n} e^{\psi_k} \tag{5.8}$$

and optimizes ψ to minimize the dev loss. Notably, unlike standard DDS we make the design decision to weight training datasets rather than score each training example $\langle x, y \rangle$ directly, as it is more efficient and also likely easier to learn.

We can thus rewrite the objective in Eq. 5.2 to incorporate both ψ and θ as:

$$\begin{aligned} \psi^* &= \operatorname*{argmin}_{\psi} J_{\text{dev}}(\theta^*(\psi), D_{\text{dev}}) \ \text{ where} \\ \theta^* &= \operatorname*{argmin}_{\theta} \mathbb{E}_{i \sim P_D(i;\psi)} \left[J(\theta, D^i_{\text{train}}) \right] \end{aligned} \tag{5.9}$$

In other words, while the general DDS framework evaluates the model performance on a single dev set and optimizes the weighting of each training example, our multilingual training objective evaluates the performance over an aggregation of n dev sets and optimizes the weighting of n training sets.

The reward signal for updating ψ_t is

$$\begin{aligned} R(i;\theta_t) &\approx \cos\left(\nabla\left(J_{\text{dev}}(\theta_t, D_{\text{dev}})\right), \nabla_\theta J(\theta_{t-1}, D^i_{\text{train}})\right) \\ &= \cos\left(\nabla\left(\frac{1}{n}\sum_{k=1}^{n} J(\theta_t, D^k_{\text{dev}})\right), \nabla_\theta J(\theta_{t-1}, D^i_{\text{train}})\right), \end{aligned} \tag{5.10}$$

where $J_{\text{dev}}(\cdot)$ defines the combination of n dev sets, and we simply plug in its definition from Eq. 5.3. Intuitively, Eq. 5.10 implies that we should favor the training language i if its gradient aligns with the gradient of the aggregated dev risk of all languages.

Implementing the Scorer Update The pseudo-code for the training algorithm using MultiDDS can be found in Alg. 3. Notably, we do not update the data scorer ψ on every training step, because it is too computationally expensive for NMT training (Wang et al., 2019c). Instead, after training the multilingual model θ for a certain number of steps, we update the scorer for all languages. This implementation is not only efficient, but also allows us to re-estimate more frequently the effect of languages that have low probability of being sampled.

In order to do so, it is necessary to calculate the effect of each training language on the current model, namely $R(i;\theta_t)$. We estimate this value by sampling a batch of data from each D^i_{train} to get the training gradient for θ_t, and use this to calculate the reward for this language. This process is detailed in line 11 of the Alg. 3.

Unlike the algorithm in DDS which requires storing n model gradients,[3] this approximation does not require extra memory even if n is large, which is important given recent efforts to scale multilingual training to 100+ (Arivazhagan et al., 2019; Aharoni et al., 2019) or even 1000+ languages (Östling and Tiedemann, 2017; Malaviya et al., 2017).

5.5 Stabilized Multi-objective Training

In our initial attempts to scale DDS to highly multi-lingual training, we found that one challenge was that the reward for updating the scorer became unstable. This is because the gradient of a multilingual dev set is less consistent and of higher variance than that of a monolingual dev set, which influences the fidelity of the data scorer reward. [4]

Thus, instead of using the gradient alignment between the training data and the aggregated loss of n dev sets as the reward, we propose a second approach to first calculate the gradient alignment reward between the data and each of the n dev sets, then take the average of these as the final reward. This can be expressed mathematically as follows:

$$\begin{aligned} R'(i;\theta_t) &\approx \cos\left(\nabla_\theta\left(\frac{1}{n}\sum_{k=1}^{n} J(\theta_t, D^k_{\text{dev}})\right), \nabla_\theta J(\theta_{t-1}, D^i_{\text{train}})\right) \\ &\approx \frac{1}{n}\sum_{k=1}^{n}\cos\left(\nabla_\theta J(\theta_t, D^k_{\text{dev}}), \nabla_\theta J(\theta_{t-1}, D^i_{\text{train}})\right) \end{aligned} \tag{5.11}$$

To implement this, we can simply replace the standard reward calculation at Line 11 of Alg. 3 to use the stable reward. We name this setting MultiDDS-S. In §5.7.1 we show that this method has less variance than the reward in Eq. 5.10.

[3] The NMT algorithm in (Wang et al., 2019c) estimates the reward by storing the moving average of n training gradients, which is not memory efficient (See Line. 7 of Alg. 2 in Chapter 4). In the preliminary experiments, our approximation performs as well as the moving average approximation. Thus, we use our approximation method as the component for MultiDDS for the rest of the experiments.

[4] Suppose the dev set gradient of language k has variance of $\text{var}(g^k_{\text{dev}}) = \sigma$, and that the dev gradients of each language $\{g^1_{\text{dev}}, ..., g^n_{\text{dev}}\}$ are independent. Then the sum of the gradients from the n languages has a variance of $\text{var}(\sum_{k=1}^{n} g^k_{\text{dev}}) = n\sigma$.

Algorithm 3: Training with MultiDDS

Input : $\mathcal{D}_{\text{train}}$; M: amount of data to train the multilingual model before updating ψ;
Output: The converged multilingual model θ^*

▷ *Initialize $P_D(i, \psi)$ to be proportional to dataset size*

$P_D(i, \psi) \leftarrow \frac{|D^i_{\text{train}}|}{\sum_{j=1}^{n} |D^j_{\text{train}}|}$

while *not converged* **do**

  ▷ *Load training data with ψ*

  $X, Y \leftarrow \emptyset$

  while $|X, Y| < M$ **do**

    $\tilde{i} \sim P_D(i, \psi_t)$

    $(x, y) \sim D^{\tilde{i}}_{\text{train}}$

    $X, Y \leftarrow X, Y \cup x, y$

  end

  ▷ *Train the NMT model for multiple steps*

  for x, y *in* X, Y **do**

    $\theta \leftarrow \text{GradientUpdate}\ (\theta, \nabla_\theta \ell(x, y; \theta))$

  end

  ▷ *Estimate the effect of each language $R(i; \theta)$*

  for *i from 1 to n* **do**

    $x', y' \sim D^i_{\text{train}}$

    $g_{\text{train}} \leftarrow \nabla_\theta \ell(x', y'; \theta)$

    $\theta' \leftarrow \text{GradientUpdate}(\theta, g_{\text{train}})$

    $g_{\text{dev}} \leftarrow 0$

    for *j from 1 to n* **do**

      $x_d, y_d \sim D^j_{\text{dev}}$

      $g_{\text{dev}} \leftarrow g_{\text{dev}} + \nabla_{\theta'} \ell(x_d, y_d; \theta')$

    end

    $R(i; \theta) \leftarrow \cos(g_{\text{dev}}, g_{\text{train}})$

  end

  ▷ *Optimize ψ*

  $d_\psi \leftarrow \sum_{i=1}^{n} R(i; \theta) \cdot \nabla_\psi \log\left(P_D\ (i; \psi)\right)$

  $\psi \leftarrow \text{GradientUpdate}(\psi, d_\psi)$

end

5.6 Experimental Evaluation

5.6.1 Data and Settings

We use the 58-languages-to-English parallel data from Qi et al. (2018). A multilingual NMT model is trained for each of the two sets of language pairs with different level of language diversity:

Related: 4 LRLs (Azerbaijani: `aze`, Belarusian: `bel`, Glacian: `glg`, Slovak: `slk`) and a related HRL for each LRL (Turkish: `tur`, Russian: `rus`, Portuguese: `por`, Czech: `ces`)

Diverse: 8 languages with varying amounts of data, picked without consideration for relatedness (Bosnian: `bos`, Marathi: `mar`, Hindi: `hin`, Macedonian: `mkd`, Greek: `ell`, Bulgarian: `bul`, French: `fra`, Korean: `kor`)

For each set of languages, we test two varieties of translation: 1) many-to-one (M2O): translating 8 languages to English; 2) one-to-many (O2M): translating English into 8 different languages. A target language tag is added to the source sentences for the O2M setting (Johnson et al., 2016b).

5.6.2 Experiment Setup

All translation models use standard transformer models Vaswani et al. (2017) as implemented in fairseq (Ott et al., 2019) with 6 layers and 4 attention heads. All models are trained for 40 epochs. We preprocess the data using sentencpiece (Kudo and Richardson, 2018) with a vocabulary size of $8K$ for each language. The model performance is evaluated with BLEU score (Papineni et al., 2002), using sacreBLEU (Post, 2018).

Baselines We compare with the three standard heuristic methods explained in §5.2: 1) Uniform ($\tau = \infty$): datasets are sampled uniformly, so that LRLs are over-sampled to match the size of the HRLs; 2) Temperature: scales the proportional distribution by $\tau = 5$ (following Arivazhagan et al. (2019)) to slightly over-sample the LRLs; 3) Proportional ($\tau = 1$): datasets are sampled proportional to their size, so that there is no over-sampling of the LRLs.

Ours we run MultiDDS with either the standard reward (MultiDDS), or the stabilized reward proposed in Eq. 5.11 (MultiDDS-S). The scorer for MultiDDS simply maps the ID of each dataset to its corresponding probability (See Eq. 5.8. The scorer has N parameters for a dataset with N languages.)

	Method	**Avg.**	**aze**	**bel**	**glg**	**slk**	**tur**	**rus**	**por**	**ces**
M2O	Prop.	24.88	11.20	17.17	27.51	28.85	**23.09***	**22.89**	**41.60**	26.80
	MultiDDS-S	**25.52**	**12.20***	**19.11***	**29.37***	**29.35***	22.81	22.78	41.55	**27.03**
O2M	Temp.	16.61	**6.66**	11.29	**21.81**	18.60	11.27	14.92	32.10	16.26
	MultiDDS-S	**17.32**	6.59	**12.39***	21.65	**20.61***	**11.58**	**15.26***	**33.52***	**16.98***
			bos	**mar**	**hin**	**mkd**	**ell**	**bul**	**fra**	**kor**
M2O	Prop.	26.68	23.43	10.10	22.01	31.06	**35.62***	**36.41***	**37.91***	**16.91**
	MultiDDS-S	**27.00**	**25.34***	**10.57**	**22.93***	**32.05***	35.27	35.77	37.30	16.81
O2M	Temp.	17.94	**14.73***	**4.93**	15.49	20.59	24.82	26.60	**29.74***	6.62
	MultiDDS-S	**18.24**	14.02	4.76	**15.68***	**21.44**	**25.69***	**27.78***	29.60	**7.01***

Table 5.1: BLEU scores of the best baseline and MultiDDS-S for all translation settings. MultiDDS-S performs better on more languages. For each setting, bold indicates the highest value, and $*$ means the gains are statistically significant with $p < 0.05$.

	Method	M2O		O2M	
		Related	**Diverse**	**Related**	**Diverse**
Baseline	Uni. (τ=∞)	22.63	24.81	15.54	16.86
	Temp. (τ=5)	24.00	26.01	16.61	17.94
	Prop. (τ=1)	24.88	26.68	15.49	16.79
Ours	MultiDDS	25.26	26.65	17.17	**18.40**
	MultiDDS-S	**25.52**	**27.00**	**17.32**	18.24

Table 5.2: Average BLEU for the baselines and our methods. Bold indicates the highest value.

5.6.3 Main Results

We first show the average BLEU score over all languages for each translation setting in Tab. 5.2. First, comparing the baselines, we can see that there is no consistently strong strategy for setting the sampling ratio, with proportional sampling being best in the M2O setting, but worst in the O2M setting. Next, we can see that MultiDDS outperforms the best baseline in three of the four settings and is comparable to proportional sampling in the last M2O-Diverse setting. With the stabilized reward, MultiDDS-S consistently delivers better overall performance than the best baseline, and outperforms MultiDDS in three settings. From these results, we can conclude that MultiDDS-S provides a stable strategy to train multilingual systems over a variety of settings.

Next, we look closer at the BLEU score of each language pair for MultiDDS-S and the best baseline. The results for all translation settings are in Tab. 5.1. In general, MultiDDS-S outperforms the baseline on more languages. In the best case, for the O2M-Related setting, MultiDDS-S brings significant gains for five of the eight languages, without hurting the remaining three. The gains for the Related group are larger than for the Diverse group, likely because MultiDDS can take better advantage of language similarities than the baseline methods.

It is worth noting that MultiDDS does not impose large training overhead. For example, for our M2O system, the standard method needs around 19 hours and MultiDDS needs around 20 hours for convergence. The change in training time is not siginificant because MultiDDS only optimizes a simple distribution over the training datasets.

5.6.4 Prioritizing what to Optimize

Prior works on multilingual models generally focus on improving the average performance of the model on all supported languages (Arivazhagan et al., 2019; Conneau et al., 2019). The formulation of MultiDDS reflects this objective by defining the aggregation of n dev sets using Eq. 5.3, which is simply the average of dev risks. However, average performance might not be the most desirable objective under all practical usage settings. For example, it may be desirable to create a more *egalitarian* system that performs well on all languages, or a more *specialized* system that does particularly well on a subset of

Setting	Baseline	MultiDDS-S		
		Regular	Low	High
M2O	26.68	27.00	26.97	27.08
O2M	17.94	18.24	17.95	18.55

Table 5.3: Average BLEU of the best baseline and three MultiDDS-S settings for the Diverse group. MultiDDS-S always outperform the baseline.

languages.

In this section, we examine the possibility of using MultiDDS to control the priorities of the multilingual model by defining different dev set aggregation methods that reflect these priorities. To do so, we first train the model for 10 epochs using regular MultiDDS, then switch to a different dev set aggregation method. Specifically, we compare MultiDDS with three different priorities:

Regular: this is the standard MultiDDS that optimizes all languages throughout training using the average dev risk aggregation in Eq. 5.3

Low: a more egalitarian system that optimizes the average of the four languages with the worst dev perplexity, so that MultiDDS can focus on optimizing the low-performing languages

High: a more specialized system that optimizes the four languages with the best dev perplexity, for MultiDDS to focus on optimizing the high-performing languages

We performed experiments with these aggregation methods on the Diverse group, mainly because there is more performance trade-off among these languages. First, in Tab. 5.3 we show the average BLEU over all languages, and find that MultiDDS with different optimization priorities still maintains competitive average performance compared to the baseline.

More interestingly, in Fig. 5.2, we plot the BLEU score difference of High and Low compared to Regular for all 8 languages. The languages are ordered on the x-axis from left to right in decreasing perplexity. Low generally performs better on the low-performing languages on the left, while High generally achieves the best performance on the high-performing languages on the right, with results most consistent in the O2M setting. This indicates that MultiDDS is able to prioritize different predefined objectives.

It is also worth noting that low-performing languages are not always low-resource languages. For example, Korean (kor) has the largest amount of training data, but its BLEU score is among the lowest. This is because it is typologically very different from English and the other training languages. Fig. 5.2 shows that Low is still able to focus on improving kor, which aligns with the predefined objective. This fact is not considered in baseline methods that only consider data size when sampling from the training datasets.

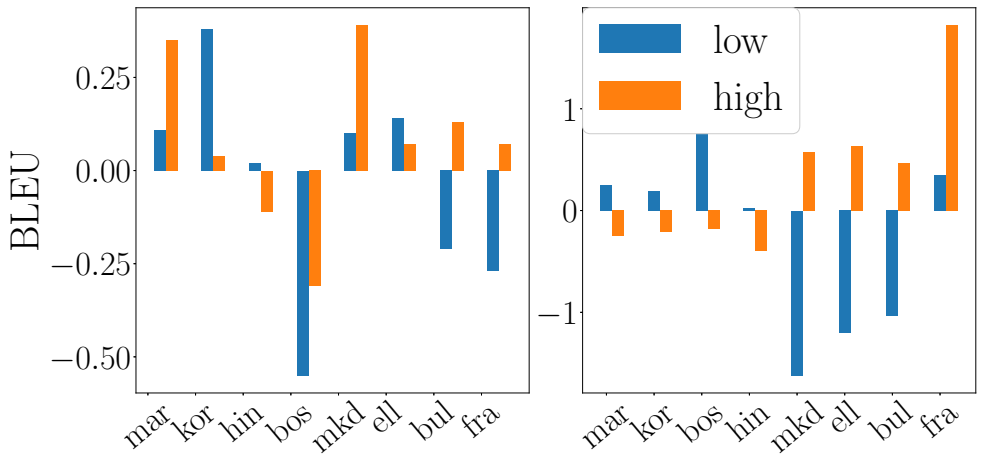

Figure 5.2: The difference between Low and High optimization objectives compared to Regular for the Diverse language group. MultiDDS successfully optimize for different priorities. *left*: M2O; *right*: O2M.

Method	**M2O**		**O2M**	
	Mean	**Var.**	**Mean**	**Var.**
MultiDDS	26.85	0.04	18.20	0.05
MultiDDS-S	26.94	0.02	18.24	0.02

Table 5.4: Mean and variance of the average BLEU score for the Diverse group. The models trained with MultiDDS-S perform better and have less variance.

5.7 Analysis

5.7.1 Effect of Stablized Rewards

Next, we study the effect of the stablized reward proposed in §5.2. In Fig. 5.3, we plot the regular reward (used by MultiDDS) and the stable reward (used by MultiDDS-S) throughout training. For all settings, the reward in MultiDDS and MultiDDS-S follows the similar trend, while the stable reward used in MultiDDS-S has consistently less variance.

MultiDDS-S also results in smaller variance in the final model performance. We run MultiDDS and MultiDDS-S with 4 different random seeds, and record the mean and variance of the average BLEU score. Tab. 5.4 shows results for the Diverse group, which indicate that the model performance achieved using MultiDDS-S has lower variance and a higher mean than MultiDDS.

Additionally, we compare the learned language distribution of MultiDDS-S and MultiDDS in Fig. 5.4. The learned language distribution in both plots fluctuates similarly, but MultiDDS has more drastic changes than MultiDDS-S. This is also likely due to the reward of MultiDDS-S having less variance than that of MultiDDS.

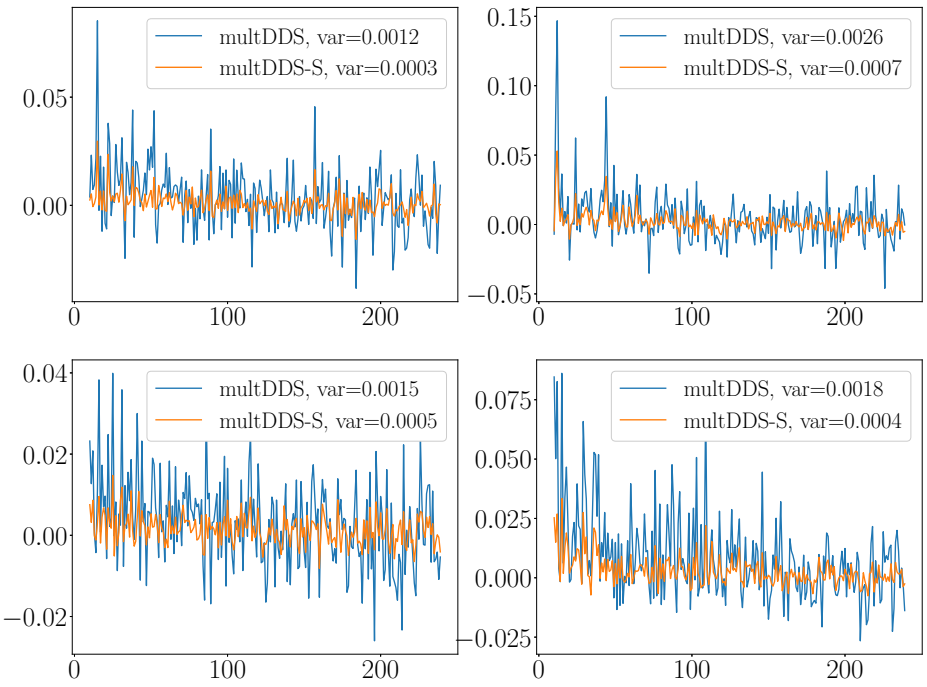

Figure 5.3: Variance of reward. *Left*: M2O; *Right*: O2M; *Top*: Related language group; *Bottom*: Diverse language group.

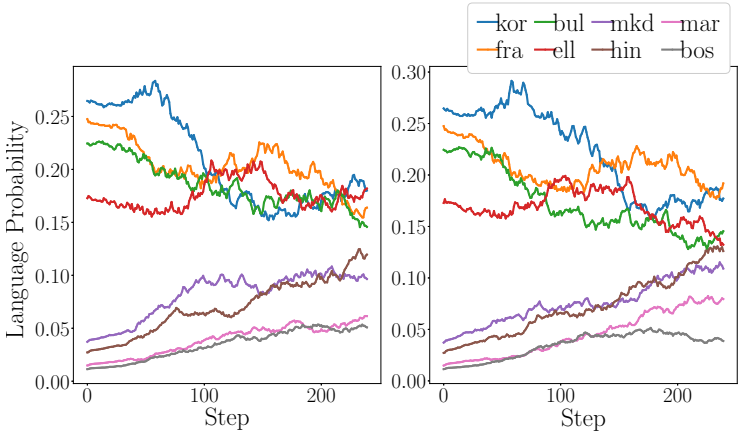

Figure 5.4: Language usage for the M2O-Diverse setting. *Left*: MultiDDS-S; *Right*: MultiDDS. The two figures follow similar trends while MultiDDS changes more drastically.

5.7.2 Learned Language Distributions

In Fig. 5.5, we visualize the language distribution learned by MultiDDS throughout the training process. Under all settings, MultiDDS gradually increases the usage of LRLs. Although initialized with the same distribution for both one-to-many and many-to-one settings, MultiDDS learns to up-sample the LRLs more in the one-to-many setting, likely due to the increased importance of learning language-specific decoders in this setting. For the Diverse group, MultiDDS learns to decrease the usage of Korean (kor) the most, probably because it is very different from other languages in the group.

5.8 Related Work

Our work is related to the multilingual training methods in general. As stated previously, recent results have demonstrated the importance of balancing HRLs and LRLs during multilingual training (Arivazhagan et al., 2019; Conneau et al., 2019), which is largely done with heuristic sampling using a temperature term; MultiDDS provides a more effective and less heuristic method. Wang and Neubig (2019); Lin et al. (2019) choose languages from multilingual data to improve the performance on a particular language, while our work instead aims to train a single model that handles translation between many languages. (Zaremoodi et al., 2018; Wang et al., 2018b, 2019b) propose improvements to the model architecture to improve multilingual performance, while MultiDDS is a model-agnostic and optimizes multilingual data usage.

Our work is also related to machine learning methods that balance multitask learning (Chen et al., 2018; Kendall et al., 2018). For example, Kendall et al. (2018) proposes to weigh the training loss from a multitask model based on the uncertainty of each task. Our method focuses on optimizing the multilingual data usage, and is both somewhat orthogonal to and less heuristic than such loss weighting methods. Finally, our work is related to meta-learning, which is used in hyperparameter optimization (Baydin et al., 2018), model initialization for fast adaptation (Finn et al., 2017), and data weighting (Ren et al., 2018). Notably, Gu et al. (2018b) apply meta-learning to learn an NMT model initialization for a set of languages, so that it can be quickly fine-tuned for any language. This is different in motivation from our method because it requires an adapted model for each of the language, while our method aims to optimize a single model to support all languages. To our knowledge, our work is the first to apply meta-learning to optimize data usage for multilingual objectives.

5.9 Conclusion

In this chapter, we propose MultiDDS, an algorithm that learns a language scorer to optimize multilingual data usage to achieve good performance on many different languages. We extend and improve over the method DDS (Wang et al., 2019c) introduced in Chapter 4, with a more efficient algorithmic instantiation tailored for the multilingual training problem and a stable reward to optimize multiple objectives. MultiDDS not only outperforms prior methods in terms of overall performance on all languages, but

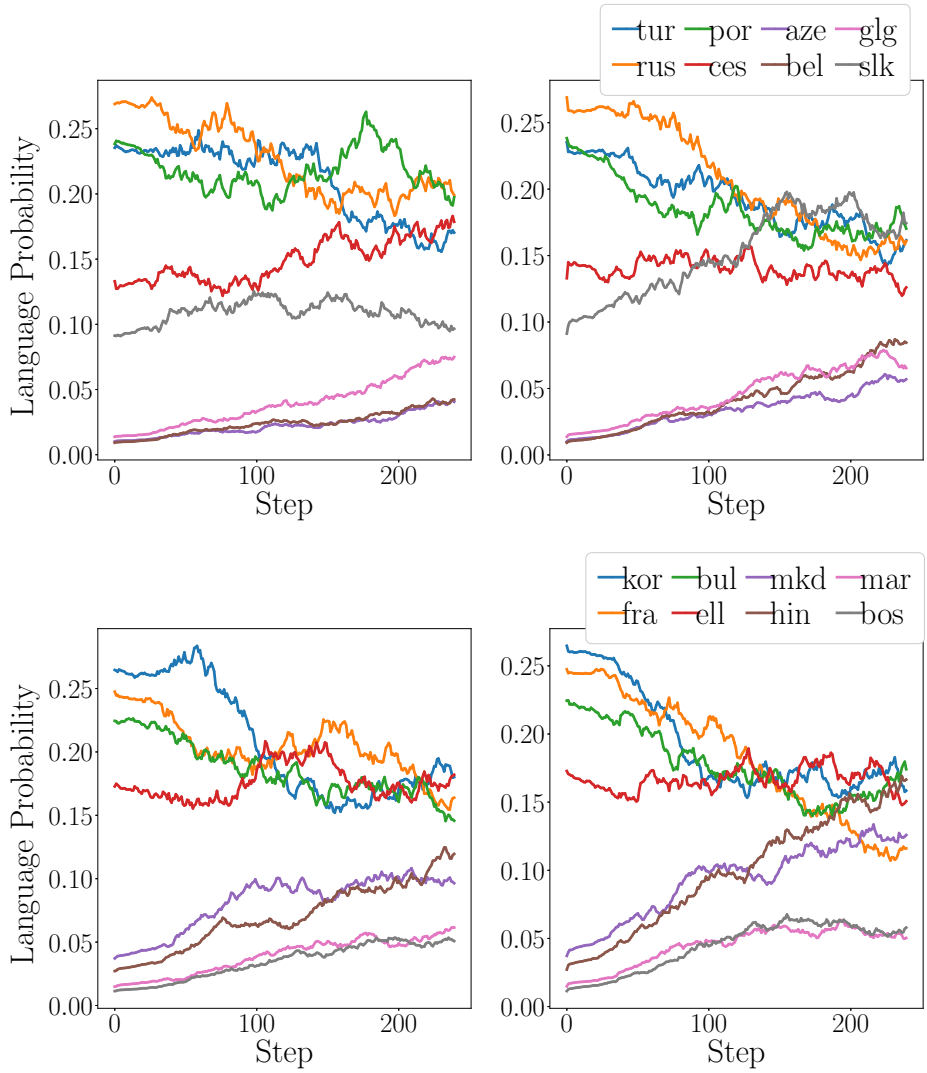

Figure 5.5: Language usage by training step. *Left*: many-to-one; *Right*: one-to-many; *Top*: related language group; *Bottom*: diverse language group.

also provides a flexible framework to prioritize different multilingual objectives.

Notably, MultiDDS is not limited to NMT, and future work may consider applications to other multilingual tasks. In addition, there are other conceivable multilingual optimization objectives than those we explored in §5.6.4.

Part II

Data Representation

Chapter 6

Multilingual Neural Machine Translation with Soft Decoupled Encoding

After selecting suitable training examples from the multilingual dataset, it is still challenging to represent the selected multilingual data which often have diverse scripts and vocabularies. We first explore modeling improvements to address this problem for NMT. In this chapter, we propose Soft Decoupled Encoding (SDE), a multilingual lexicon encoding framework specifically designed to share lexical-level information intelligently without requiring heuristic preprocessing such as pre-segmenting the data.[1]

6.1 Introduction

The standard sequence-to-sequence (seq2seq) NMT model (Sutskever et al., 2014) represents each lexical unit by a vector from a look-up table, making it difficult to share across different languages with limited lexicon overlap. This problem is particularly salient when translating low-resource languages, where there is not sufficient data to fully train the word embeddings. Several methods have been proposed to alleviate this data sparsity problem in multilingual lexical representation. The current de-facto standard method is to use subword units (Sennrich et al., 2016b; Kudo, 2018a), which split up longer words into shorter subwords to allow for generalization across morphological variants or compounds (e.g. "un/decide/d" or "sub/word"). These can be applied to the concatenated multilingual data, producing a shared vocabulary for different languages, resulting in sharing some but not all subwords of similarly spelled words (such as "traducción" in Spanish and "tradução" in Portuguese, which share the root "tradu-"). However, subword-based preprocessing can produce sub-optimal segmentations for multilingual data, with semantically identical and similarly spelled languages being split into different granularities (e.g. "traducción" and "tradu/ção") leading to disconnect in the resulting representations. This problem is

[1]The source code is available at https://github.com/cindyxinyiwang/SDE

especially salient when the high-resource language dominates the training data (see empirical results in §8.4).

In this chapter, we propose Soft Decoupled Encoding (SDE), a multilingual lexicon representation framework that obviates the need for segmentation by representing words on a full-word level, but can nonetheless share parameters intelligently, aiding generalization. Specifically, SDE *softly* decouples the traditional word embedding into two interacting components: one component represents how the word is spelled, and the other component represents the word's latent meaning, which is shared over all languages present at training time. We can view this representation as a decomposition of language-specific realization of the word's form (i.e. its spelling) and its language-agnostic semantic function. More importantly, our decoupling is done in a *soft* manner to preserve the interaction between these two components.

SDE has three key components: 1) an encoding of a word using character n-grams (Wieting et al., 2016); 2) a language specific transform for the character encoding; 3) a latent word embedding constructed by using the character encoding to attend to a shared word embedding space, inspired by Gu et al. (2018a). Our method can enhance lexical-level transfer through the shared latent word embedding while preserving the model's capacity to learn specific features for each language. Moreover, it eliminates unknown words without any external preprocessing step such as subword segmentation.

We test SDE on four low-resource languages from a multilingual TED corpus (Qi et al., 2018). Our method shows consistent improvements over multilingual NMT baselines for all four languages, and importantly outperforms previous methods for multilingual NMT that allow for more intelligent parameter sharing but do not use a two-step process of character-level representation and latent meaning representation (Gu et al., 2018a). Our method outperforms the best baseline by about 2 BLEU for one of the low-resource languages, achieving new state-of-the-art results on all four language pairs compared to strong multi-lingually trained and adapted baselines (Neubig and Hu, 2018).

6.2 Lexical Representation for Multilingual NMT

In this section, we first revisit the 3-step process of computing lexical representations for multilingual NMT, which is illustrated in Fig. 6.1. Then, we discuss various design choices for each step, as well as the desiderata of an ideal lexical representation for multilingual NMT.

6.2.1 Lexical Unit Segmentation

The first step to compute the neural representation for a sentence is to segment the sentence into lexical units. There are three popular options with different granularities:

- **Word-based** method splits an input sequence into words, often based on white spaces or punctuation. This is perhaps the natural choice for lexical unit segmentation. Early work in NMT all employ this method (Sutskever et al., 2014; Bahdanau et al., 2015).
- **Character-based** method splits an input sequence into characters (Lee et al., 2017).

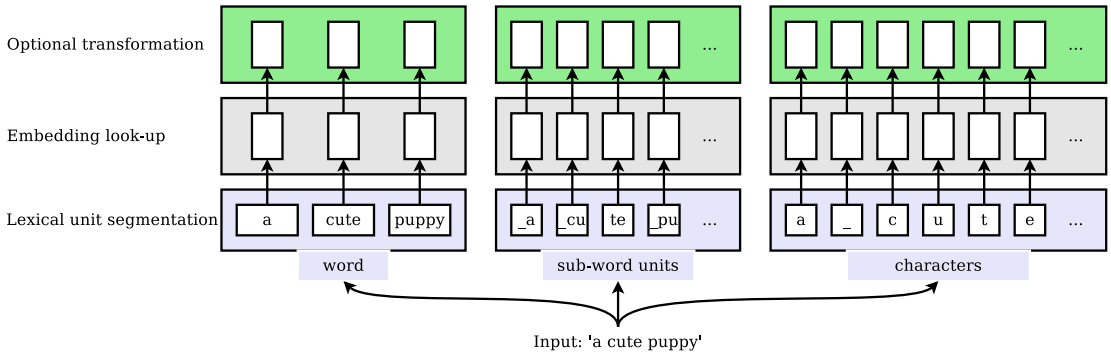

Figure 6.1: Three steps to compute the lexical representations for the input sequence "a cute puppy", using various lexical unit segmentations.

- **Subword-based** method splits each word into pieces from a small vocabulary of frequently occurring patterns (Sennrich et al., 2016b; Kudo, 2018a).

6.2.2 Embedding Look-up

After a sentence is segmented into lexical units, NMT models generally *look up* embeddings from a dictionary to turn each lexical unit into a high-dimensional vector. In the context of multilingual NMT, the lexical unit segmentation method affects this dictionary in different ways.

In **word-based** segmentation, since the number of unique words for each language is unbounded, while computer memory is not, previous work, e.g. Sutskever et al. (2014), resorts to a fixed-size vocabulary of the most frequent words, while mapping out-of-vocabulary words to an ⟨unk⟩ token. For multilingual NMT settings, where multiple languages are processed, the number of words mapped to ⟨unk⟩ significantly increases. Moreover, different languages, even related languages, have very few words that have exactly the same spelling, which leads to the same concept being represented by multiple and independent parameters. This disadvantage hurts the translation model's ability to learn the same concept in multiple languages.

Meanwhile, **character-based** segmentation can effectively reduce the vocabulary size, while maximizing the potential for parameter sharing between languages with identical or similar character sets. However, character segmentation is based on the strong assumption that neural networks can infer meaningful semantic boundaries and compose characters into meaningful words. This puts a large amount of pressure on neural models, requiring larger model sizes and training data. Additionally, training character-based NMT systems is often slow, due to the longer character sequences (Cherry et al., 2018a).

Subword-based segmentation is a middle ground between word and character segmentation. However, in multilingual translation, the subword segmentation can be sub-optimal, as the subwords from high-resource languages, i.e. languages with more training data, might dominate the subword vocabulary, so that the words in low-resource language can be split into extremely small pieces.

Therefore, existing methods for lexical unit segmentation lead to difficulties in building an effective embedding look-up strategy for multilingual NMT.

6.2.3 Optional Encoding Transformations

Most commonly, the embedding vectors looked up from the embedding table are used as the final lexical representation. However, it is also possible to have multiple versions of embeddings for a single lexicon and combine them through operations such as attentional weighted sum (Gu et al., 2018a). Without loss of generality, we can assume there is always a transformation applied to the embedding vectors, and models that do not use such a transformation can be treated as using the identity transformation.

6.2.4 Desiderata

To efficiently utilize parameters for multilingual NMT, the lexical representation should have two properties. First, for maximal accuracy, the lexical representation should be able to *accurately represent words in all of the languages under consideration.* Second, for better cross-lingual learning and generalization, such a representation should *maximize the sharing of parameters across languages.*

These two conflicting objectives are difficult to achieve through existing methods. The most common method of using lookup embeddings can only share information through lexical units that overlap between the languages. Subword segmentation strikes a middle ground, but has many potential problems for multilingual NMT, as already discussed in §6.2.2. Although Gu et al. (2018a)'s method of latent encoding increases lexical level parameter sharing, it still relies on subwords as its fundamental units, and thus inherits the previously stated problems of sub-word segmentation. We also find in experiments in §8.4 that it is actually *less* robust than simple lookup when large monolingual data to pre-train embeddings is not available, which is the case for many low-resourced languages.

Next, in §6.3, we propose a novel lexical representation strategy that achieves both desiderata.

6.3 Soft Decoupled Encoding

Given the conflict between sharing lexical features and preserving language specific properties, we propose SDE, a general framework to represent lexical units for multilingual NMT. Specifically, following the linguistic concept of the "arbitrariness of the sign" (Chandler, 2007), SDE decomposes the modeling of each word into two stages: (1) modeling the language-specific spelling of the word, and (2) modeling the language-agnostic semantics of the word. This decomposition is based on the need for distinguished treatments between a word's semantics and its spelling.

Semantic representation is language-agnostic. For example, the English "hello" and the French "bonjour" deliver the same greeting message, which is invariant with respect to the language. SDE shares such semantic representations among languages by querying a list of shared concepts, which are loosely related to the linguistic concept of "sememes" (Greimas, 1983). This design is implemented using an

Method	Lex Unit	Embedding	Encoding
Johnson et al. (2016a)	Subword	joint-Lookup	Identity
Lee et al. (2017)	Character	joint-Lookup	Identity
Gu et al. (2018a)	Subword	pretrain-Lookup	joint-Lookup + Latent
Ataman and Federico (2018)	Word	character n-gram	Identity
SDE	Word	character n-gram	Identity + Latent

Table 6.1: Methods for lexical representation in multilingual NMT. *joint-Lookup* means the lookup table is jointly trained with the whole model. *pretrain-Lookup* means the lookup table is trained independently on monolingual data.

attention mechanism, where the query is the lexical unit representation, and the keys and the values come from an embedding matrix shared among all languages.

Meanwhile, the **word spellings** are more sophisticated. Here, we identify two important observations about word spellings. First, words in related languages can have similar spellings, e.g. the English word "color" and the French word "coleur". In order to effectively share parameters among languages, a word spelling model should utilize this fact. Second, and not contradicting the first point, related languages can also exhibit consistent spelling shifts. For instance, "Christopher", a common name in English, has the spelling "Kryštof" in Czech. This necessitates a learnable rule to convert the spelling representations between such pairs of words in related languages. To account for both points, we use a language-specific transformation on top of a first encoding layer based on character n-grams.

6.3.1 Existing Methods

Before we describe our specific architecture in detail (§6.3.2), given these desiderata discussed above, we summarize the designs of several existing methods for lexical representation and our proposed SDE framework in Tab. 6.1. Without a preprocessing step of subword segmentation, SDE can capture the lexical similarities of two related languages through the character n-gram embedding while preserving the semantic meaning of lexicons through a shared latent embedding.

6.3.2 Details of Soft Decoupled Encoding

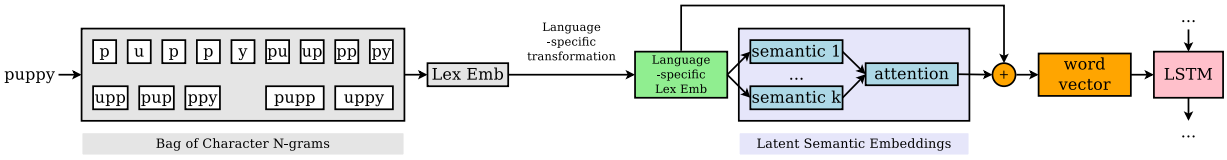

Figure 6.2: SDE computes the embedding for the word "puppy". Both character n-grams embeddings and latent semantic embeddings are shared among all languages.

As demonstrated in Fig. 6.2, given a word w in a multilingual corpus from language L_i, SDE constructs the embedding of w in three phases.

Lexical Embedding. We maintain an embedding matrix $\mathbf{W}_c \in \mathbb{R}^{C \times D}$, where D is the embedding dimension and C is the number of n-grams in our vocabulary. Out-of-vocab n-grams are mapped to a designated token $\langle \text{unk} \rangle$. $\mathbf{W}_c$ is shared among all languages. Following Wieting et al. (2016), for each word w, we first compute the bag of character n-grams of w, denoted by $\text{BoN}(w)$, which is a sparse vector whose coordinates are the appearance counts of each character n-gram in w. For instance, the characters n-grams with $n = 1, 2, 3, 4$ of the word "puppy" are shown in Fig. 6.2. We then look up and add the rows of $\mathbf{W}_c$ according to their corresponding counts, and apply a $\tanh$ activation function on the result

$$c(w) = \tanh(\text{BoN}(w) \cdot \mathbf{W}_c). \tag{6.1}$$

Language-specific Transformation. To account for spelling shifts between languages (c.f. §6.3), we apply an language-specific transformation to normalize away these differences. We use a simple fully-connected layer for this transformation. In particular, for language L_i, we have

$$c_i(w) = \tanh(c(w) \cdot \mathbf{W}_{L_i}), \tag{6.2}$$

where $\mathbf{W}_{L_i} \in \mathbb{R}^{D \times D}$ is the transformation matrix specific to language L_i.

Latent Semantic Embedding. Finally, to model the shared semantics of words among languages, we employ an embedding matrix $\mathbf{W}_s \in \mathbb{R}^{S \times D}$, where S is the number of core semantic concepts we assume a language can express. Similar to the lexical embedding matrix $\mathbf{W}_c$, the semantic embedding matrix $\mathbf{W}_s$ is also shared among all languages.

For each word w, its language-specific embedding $c_i(w)$ is passed as a query for an attention mechanism (Luong et al., 2015) to compute a weighted sum over the latent embeddings

$$e_{\text{latent}}(w) = \text{Softmax}(c_i(w) \cdot \mathbf{W}_s^\top) \cdot \mathbf{W}_s. \tag{6.3}$$

Finally, to ease the optimization of our model, we follow Vaswani et al. (2017) and add the residual connection from $c_i(w)$ into $e_{\text{latent}}(w)$, forming the Soft Decoupled Encoding embedding of w

$$e_{\text{SDE}}(w) = e_{\text{latent}}(w) + c_i(w). \tag{6.4}$$

6.4 Experiment

We build upon a standard seq2seq NMT model for all experiments. We run each experiment with 3 different random seeds, and conduct significance tests for the results using the paired bootstrap (Clark et al., 2011a).

6.4.1 Datasets

We use the 58-language-to-English TED corpus for experiments. Following the settings of prior works on multilingual NMT (Neubig and Hu, 2018; Qi et al., 2018), we use three low-resource language datasets:

LRL	Train	Dev	Test	HRL	Train
aze	5.94k	671	903	tur	182k
bel	4.51k	248	664	rus	208k
glg	10.0k	682	1007	por	185k
slk	61.5k	2271	2445	ces	103k

Table 6.2: Statistics of our datasets. LRL and HRL mean Low-Resource and High-Resource Language.

Azerbaijani (aze), Belarusian (bel), Galician (glg) to English, and a slightly higher-resource dataset, namely Slovak (slk) to English. Each low-resource language is paired with a related high-resource language: Turkish (tur), Russian (rus), Portuguese (por), and Czech (ces) respectively. Tab. 6.2 shows the statistics of each dataset.

6.4.2 Baselines

For the baseline, we use the standard lookup embeddings for three granularities of lexical units: (1) word: with a fixed word vocabulary size of 64,000 for the concatenated bilingual data; (2) sub-joint: with BPE of 64,000 merge operations on the concatenated bilingual data; and (3) sub-sep: with BPE separately on both languages, each with 32,000 merge operations, effectively creating a vocabulary of size 64,000. We use all three settings to compare their performances and to build a competitive baselines. We also implement the latent embedding method of Gu et al. (2018a). We use 32,000 character n-gram with $n = \{1, 2, 3, 4, 5\}$ from each language and a latent embedding size of 10,000.

6.4.3 Results

Table 6.3 presents the results of SDE and of other baselines. For the three baselines using lookup, sub-sep achieves the best performance for three of the four languages. Sub-joint is worse than sub-sep although it allows complete sharing of lexical units between languages, probably because sub-joint leads to over-segmentation for the low-resource language. Our reimplementation of universal encoder (Gu et al., 2018a) does not perform well either, probably because the monolingual embedding is not trained on enough data, or the hyperparameters for their method are harder to tune. Meanwhile, SDE outperforms the best baselines for all four languages, without using subword units or extra monolingual data.

[2]For all tasks in our experiments, our reimplementation of Neubig and Hu (2018) achieves similar or slightly higher BLEU scores than originally reported (aze: 10.9; bel: 15.8; glg: 27.3; slk: 25.5). We suspect the difference is because we use a different tokenizer from Neubig and Hu (2018). Details are in our open-sourced software.

[3]To ensure the fairness of comparison with other methods, we only train the monolingual embedding from the parallel training data, while Gu et al. (2018a) used extra monolingual data from the Wikipedia dump. We have also tried testing our reimplementation of their method with trained monolingual embedding from the Wikipedia dump, but achieved similar performance.

Lex Unit	Model	aze	bel	glg	slk
Word	Lookup	7.66	13.03	28.65	25.24
Sub-joint	Lookup	9.40	11.72	22.67	24.97
Sub-sep	Lookup (Neubig and Hu, 2018)[2]	10.90	16.17	28.10	28.50
Sub-sep	UniEnc (Gu et al., 2018a)[3]	4.80	8.13	14.58	12.09
Word	SDE	11.82*	18.71*	30.30*	28.77†

Table 6.3: BLEU scores on four language pairs. Statistical significance is indicated with $*$ ($p < 0.0001$) and † ($p < 0.05$), compared with the best baseline.

Model	aze	bel	glg	slk
SDE	11.82	18.71	30.30	28.77
-Language Specific Transform	12.89*	18.13†	30.07	29.16†
-Latent Semantic Embedding	7.77*	15.66*	29.25*	28.15*
-Lexical Embedding	4.57*	8.03*	13.77*	7.08*

Table 6.4: BLEU scores after removing each component from SDE-com. Statistical significance is indicated with $*$ ($p < 0.0001$) and † ($p < 0.005$), compared with the full model in the first row.

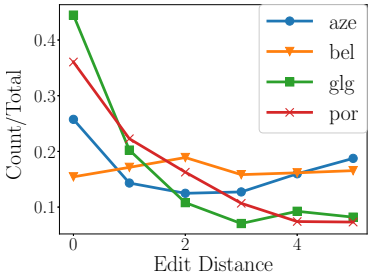

Figure 6.3: Percentage of words by the edit distance from the matching words in the high-resource language.

6.4.4 Ablation Studies

We next ablate various features of SDE by removing each of the three key components, and show the results in Tab. 7.3. Removing the latent semantic embedding and lexical embedding consistently harms the performance of the model. The effect of the language specific transformation is smaller and is language dependent. Removing the language specific transformation does not lead to significant difference for glg. However, it leads to a 0.8 gain in BLEU for aze, a 0.3 gain for slk, but about a 0.6 decrease for bel. A further inspection of the four language pairs shows that the language specific transform is more helpful for training on languages with fewer words of the same spelling. To quantify this, we extract bilingual dictionaries of the low-resource languages and their paired high-resource languages, and measure the edit distance between the character strings of the aligned words. We use FastAlign (Dyer et al., 2013) to extract aligned words between the source language and English from the parallel training data, then match the English side of two related languages to get their dictionary. Tab. 6.3 shows the percentage of the word pairs grouped by their edit distance. Among the four languages, bel has the lowest percentage of words that are exactly the same with their corresponding words in the high-resource language (0 edit distance), indicating that the language specific transform is most important for divergent languages.

In the published paper corresponding to this chapter (Wang et al., 2019c), there are several additional

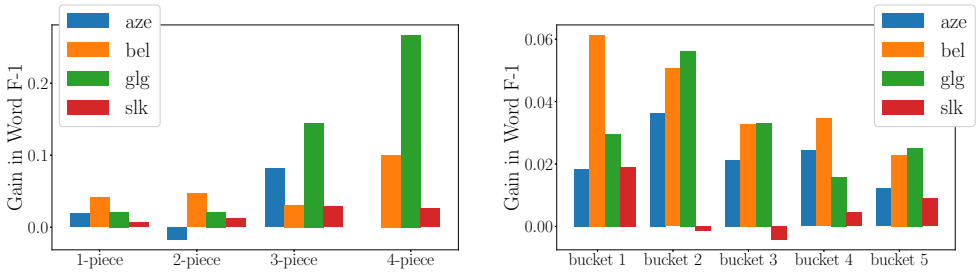

Figure 6.4: Gain in word F-measure of SDE over sub-sep. *Left*: the target words are bucketed by the number of subword pieces that their corresponding source words are segmented into. *Right*: the target words are bucketed by the edit distance between their source words and the corresponding words in the high resource language.

experiments on using SDE with subwords and models trained on all eight languages. These will be added to the final thesis document.

6.5 Analysis

6.5.1 Why does SDE work better?

The SDE framework outperforms the strong sub-sep baseline because it avoids sub-optimal segmentation of the multilingual data. We further inspect the improvements by calculating the word F-measure of the translated target words based on two properties of their corresponding source words: 1) the number of subwords they were split into; 2) the edit distance between their corresponding words in the related high-resource language. From Fig. 6.4 *left*, we can see that SDE is better at predicting words that were segmented into a large number of subwords. Fig. 6.4 *right* shows that the gain peaks on the second bucket for 2 languages and the first bucket for bel and slk, which implies that SDE shows more improvements for words with small but non-zero edit distance from the high-resource language. This is intuitive: words similar in spelling but with a few different characters can be split into very different subword segments, while SDE can leverage the lexical similarities of word pairs with slightly different spelling.

6.5.2 Qualitative Analysis

Tab. 6.5 lists a few translations for both sub-sep and SDE. We can see that SDE is better at capturing functional words like “if” and “would”. Moreover, it translates “climatologist” to a related word “weather”, probably from the prefix “climat” in glg, while sub-sep gives the totally unrelated translation of “college friend”.

glg	eng	sub-sep	SDE
Pero non temos a tecnoloxía para resolver iso, temos?	But we don't have a technology to solve that, right?	But we don't have the technology to solve that , we have?	But we don't have the technology to solve that, do we?
Se queres saber sobre o clima, preguntas a un climatólogo.	If you want to know about climate, you ask a climatologist.	If you want to know about climate, you're asking a college friend.	If you want to know about climate, they ask for a weather.
Non é dicir que si tivesemos todo o diñeiro do mundo, non o quereríamos facer.	It's not to say that if we had all the money in the world, we wouldn't want to do it .	It's not to say that we had all the money in the world, we didn't want to do it .	It's not to say that if we had all the money in the world, we wouldn't want to do it.

Table 6.5: Examples of glg to eng translations.

6.5.3 Effect of Vocabulary Size

We examine the performance of SDE and the best baseline sub-sep, with a character *n*-gram vocabulary and sub-word vocabulary respectively, of size of 8K, 16K, and 32K. We use character n-gram of $n = \{1, 2, 3, 4\}$ for 8K and 16K vocabularies, and $n = \{1, 2, 3, 4, 5\}$ for the 32K vocabulary. Figure 6.5 shows that for all four languages, SDE outperforms sub-sep with all three vocabulary sizes. This shows that SDE is also competitive with a relatively small character n-gram vocabulary.

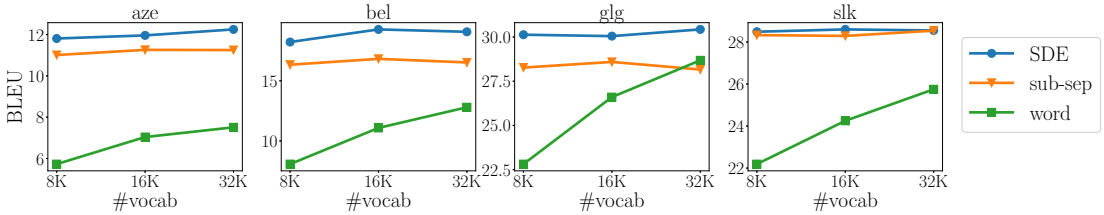

Figure 6.5: Performance on three different vocabulary size (results of a single random seed).

6.5.4 Example Bilingual Words

Table 6.6 lists some words and their subwords from bel and its related language rus. We can see that subwords fail to capture all the lexical similarities between these words, and sometimes the word pairs are segmented into different number of pieces.

6.5.5 Analysis of Attention over Latent Embedding Space

In this section we compare the attention distribution over the latent embedding space of related languages, with the intuition that words that mean the same thing should have similar attention distributions. We calculate the KL divergence of the attention distribution for word pairs in both the LRL and HRL. Figure 6.6 shows that the lowest KL divergence is generally on the diagonals representing words with identical meanings, which indicates that similar words from two related languages tend to have similar attention over the latent embedding space. Note that this is the case even for words with

bel		rus		eng
word	subword	word	subword	
фінансавыя	фінансавы я	финансовых	финансовы х	financial
стадыён	стады ён	стадион	стадион	stadium
розных	розны х	разных	разны х	different
паказаць	паказа ць	показать	показать	show

Table 6.6: Bilingual word pairs and their subword pieces.

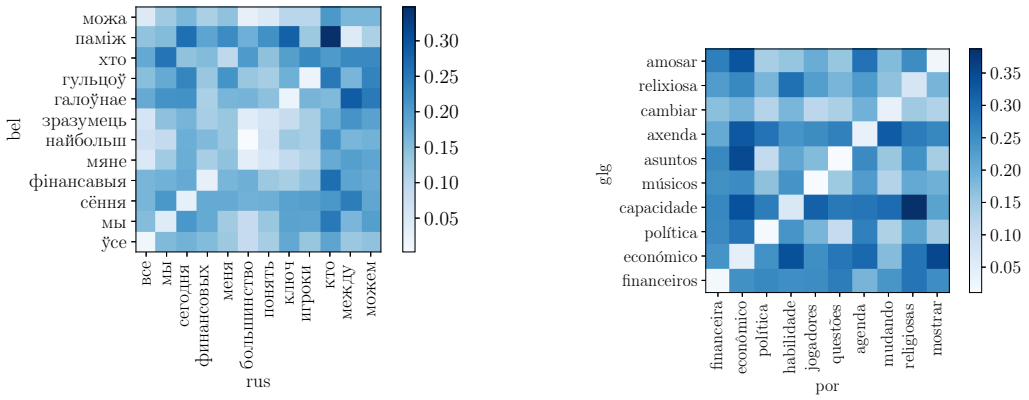

Figure 6.6: KL divergence of attention over latent embedding space between words from two related languages. Word pairs that match at the diagonal have similar meanings. *Left*: bel-rus. *Right*: glg-por.

different spellings. For example, Figure 6.6 *right* shows that the KL divergence between the glg word "músicos" (meaning "musicians"), and the por word of the closet meaning among the words shown here, "jogadores" (meaning "players"), is the smallest, although their spellings are quite different.

6.5.6 Analysis of Word Vectors from SDE

In this section, we qualitatively examine the location of word vectors at different stages of SDE. We reduce the word vectors to two dimensions using t-SNE (van der Maaten and Hinton, 2008). In particular, we focus on two groups of words shown in Table 6.7 from glg-por, where each word from glg is paired with two words from por, one with the same meaning but different spelling, while the other has similar spelling but different meaning. Figure 6.7 *left* shows the embeddings derived from the character *n*-grams. At this stage, the word "cando" is closer to "caindo", which has a different meaning, than "quando", which has the same meaning but a slightly more different spelling. The word "etiqueta" lies in the middle of "rótulo" and "riqueza". After the whole encoding process, the location of the words are shown in Figure 6.7 *right*. At the final stage, the word "cando" moves closer to "quando", which has the same meaning, than "caindo", which is more similar in spelling. The word "etiqueta" is also much closer to "rótulo", the

glg	eng	por	eng
cando	when	quando	when
		caindo	failing down
etiqueta	label	rótulo	label
		riqueza	wealth

Table 6.7: Words in glg-por that have the same meaning but different spelling, or similar spelling but different meaning.

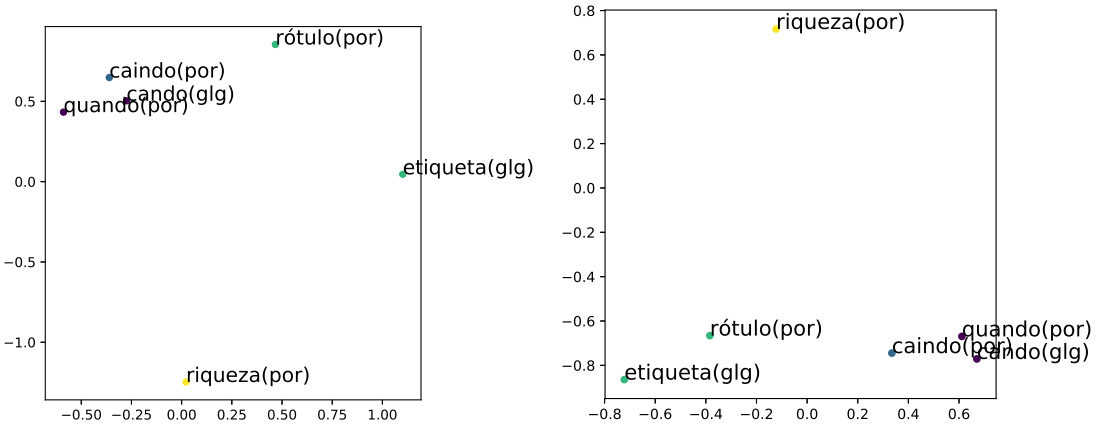

Figure 6.7: T-SNE visualizations of the embeddings of words in Table 6.7 encoded after the character *n*-gram embedding stage (*Left*), or after the full process of SDE(*Right*). Words of the same color have similar meanings, and the language code of the word is placed in the parenthesis. It can be seen that the character embedding stage is more sensitive to lexical similarity, while the full SDEmodel is more sensitive to similarity in meaning.

word with similar meaning, and grows further apart from "riqueza".

6.6 Related Works

In multilingual NMT, several approaches have been proposed to enhance parameter sharing of lexical representations. Zoph et al. (2016) randomly assigns embedding of a pretrained NMT model to the vocabulary of the language to adapt, which shows improvements over retraining the new embeddings from scratch. Nguyen and Chiang (2018) propose to match the embedding of the word piece that overlaps with the vocabulary of the new language. When training directly on concatenated data, it is also common to have a shared vocabulary of multilingual data (Neubig and Hu, 2018; Qi et al., 2018). Gu et al. (2018a) propose to enhance parameter sharing in lexical representation by a latent embedding space shared by all languages.

Several prior works have utilized character level embeddings for machine translation (Cherry et al., 2018a; Lee et al., 2017; Ataman and Federico, 2018), language modeling (Kim et al., 2016; Józefowicz et al., 2016), and semantic parsing (Yih et al., 2014). Specifically for NMT, fully character-level NMT can effectively reduce the vocabulary size while showing improvements for mulitlingual NMT (Lee et al., 2017), but it often requires much longer to train (Cherry et al., 2018a). Ataman and Federico (2018) shows that character n-gram encoding of words can improve over BPE for morphologically rich languages.

6.7 Conclusion

Existing methods of lexical representation for multilingual NMT hinder parameter sharing between words that share similar surface forms and/or semantic meanings. We show that SDE can intelligently leverage the word similarities between two related languages by softly decoupling the lexical and semantic representations of the words. Our method, used without any subword segmentation, shows significant improvements over the strong multilingual NMT baseline on all languages tested.

Chapter 7

Multi-view Subword Regularization

While designing better multilingual data representation models is a promising direction, modifying the architecture of pretrained models requires large amounts of computational resources. Similar to multilingual NMT, multilingual pretrained representations also rely on subword segmentation algorithms to create a shared multilingual vocabulary, which often lead to sub-optimal segmentation for languages with limited amounts of data. In this chapter, we want to tackle this problem for multilingual pretrained models at *fine-tuning time*.

7.1 Introduction

Multilingual pre-trained representations (Devlin et al., 2019; Huang et al., 2019; Conneau and Lample, 2019; Conneau et al., 2019) are now an essential component of state-of-the-art methods for cross-lingual transfer (Wu and Dredze, 2019; Pires et al., 2019). These methods pretrain an encoder by learning in an unsupervised way from raw textual data in up to hundreds of languages which can then be fine-tuned on annotated data of a downstream task in a high-resource language, often English, and transferred to another language. In order to encode hundreds of languages with diverse vocabulary, it is standard for such multilingual models to employ a shared subword vocabulary jointly learned on the multilingual data using heuristic word segmentation methods based on byte-pair-encoding (BPE; Sennrich et al., 2016b) or unigram language models (Kudo and Richardson, 2018) (details in §7.2). However, subword-based preprocessing can lead to sub-optimal segmentation that is inconsistent across languages, harming cross-lingual transfer performance, particularly on under-represented languages. As one example, consider the segmentation of the word "excitement" in different languages in Tab. 7.1. The English word is not segmented, but its translations in the other languages, including the relatively high-resourced French and German, are segmented into multiple subwords. Since each subword is mapped to a unique embedding vector, the segmentation discrepancy—which generally does not agree with a language's morphology—could map words from different languages to very distant representations, hurting cross-lingual transfer. In fact, previous work (Conneau et al., 2019; Artetxe et al., 2020) has shown that heuristic fixes such as increasing the subword vocabulary capacity and up-sampling low-resource languages during learning

en	excitement	**fr**	excita/tion	**de**	Auf/re/gung
pt	excita/ção	**el**	εν/θ/ουσι/ασμός	**ru**	волн/ение

Table 7.1: XLM-R segmentation of "excitement" in different languages. The English word is not segmented while the same word in other languages is over-segmented. A better segmentation would allow the model to match the verb stem and derivational affix across languages.

of the subword segmentation can lead to significant performance improvements.

Despite this, there is not much work studying or improving subword segmentation methods for cross-lingual transfer. Bostrom and Durrett (2020) empirically compare several popular word segmentation algorithms for pretrained language models of a single language. Several works propose to use different representation granularities, such as phrase-level segmentation (Zhang and Li, 2020) or character-aware representations (Ma et al., 2020) for pretrained language models of a single high-resource language, such as English or Chinese only. However, it is not a foregone conclusion that methods designed and tested on monolingual models will be immediately applicable to multilingual representations. Furthermore, they add significant computation cost to the pretraining stage, which is especially problematic for multilingual pretraining on hundreds of languages. The problem of sub-optimal subword segmentation has drawn more attention in the context of neural machine translation (NMT). Specifically, *subword regularization* methods have been proposed to improve the NMT model of a *single* language pair by randomly sampling different segmentations of the sentences during training (Kudo, 2018b; Provilkov et al., 2020). However, these methods have not been applied to multilingual NMT or pretrained language models and it is similarly not clear if they are useful for cross-lingual transfer.

In this chapter, we make two contributions to close this gap. First, we perform the first (to our knowledge) empirical examination of subword regularization methods on a variety of cross-lingual transfer tasks from the XTREME benchmark (Hu et al., 2020). We demonstrate that despite its simplicity, this method is highly effective, providing consistent improvements across a wide variety of languages and tasks for both multilingual BERT (mBERT; Devlin et al., 2019) and XLM-R (Conneau et al., 2019) models. Analysis of the results shows that this method is particularly effective for languages with non-Latin scripts despite only being applied during English fine-tuning.

Further, we posit that naively applying probabilistic segmentation only during fine-tuning may be sub-optimal as it creates a discrepancy between the segmentations during the pretraining and fine-tuning stages. To address this problem, we propose Multi-view Subword Regularization (MVR; Fig. 7.1), a novel method—inspired by the usage of consistency regularization in semi-supervised learning methods (Clark et al., 2018; Xie et al., 2018)—which utilizes *both* the standard and probabilistically segmented inputs, enforcing the model's predictions to be consistent across the two views. Such consistency regularization further improves accuracy, with MVR finally demonstrating consistent gains of up to 2.5 points over the standard practice across all models and tasks. We analyze the sources of the improvement from consistency regularization and find that it can be attributed to both label smoothing and self-ensembling.

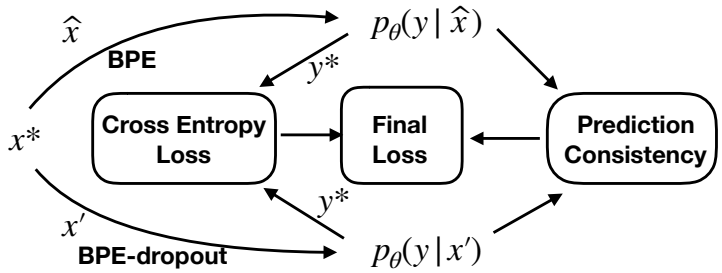

Figure 7.1: Fine-tuning models using MVR on data (x^*, y^*)

7.2 Background: Subword Segmentation

Here, we first discuss two common deterministic segmentation methods based on byte pair encoding (BPE) and unigram language models (ULM), discuss their probabilistic variants, and explain how to incorporate them in training.

7.2.1 Deterministic Segmentation

The most widely used subword segmentation methods first estimate a segmentation model from the training corpus in an unsupervised fashion. They then produce a segmentation $\widehat{x}$ of the input x^* under the estimated segmentation model $P(x)$:

$$\widehat{x} = \underset{x \in S(x^*)}{\operatorname{argmax}} P(x)$$

Here $S(x^*)$ is the set of all possible segmentations, and $P(x)$ is the likelihood of a given segmentation. Note that $\widehat{x}$ is deterministically selected for each input x^*.

Byte-pair encoding (BPE) The popular BPE algorithm (Sennrich et al., 2016b) initializes the vocabulary with individual characters and initially represents each word as a sequence of characters. It then counts the most frequent character token bigrams in the data, merges them into a new token, and adds the new token to the vocabulary. This process is done iteratively until a predefined vocabulary size is reached.

To segment a word, BPE simply splits the word into character tokens, and iteratively merges adjacent tokens with the highest priority until no merge operation is possible. That is, for an input x^*, it assigns segmentation probability $P(\widehat{x}) = 1$ for the sequence $\widehat{x}$ obtained from the greedy merge operations, and assigns other possible segmentations a probability of 0.

Notably, a variant of this method (Schuster and Nakajima, 2012) is used for the mBERT embedding model (Devlin et al., 2019).

Unigram language model (ULM) The ULM method (Kudo and Richardson, 2018) starts from a reasonably large seed vocabulary, which is iteratively pruned to maximize the training corpus likelihood under a unigram language model of the subwords until the desired vocabulary size is reached.

During segmentation, ULM decodes the most likely segmentation of a sentence under the estimated language model using the Viterbi algorithm. This method is used in the XLM-R cross-lingual embeddings (Conneau et al., 2019).

7.2.2 Probabilistic Segmentation

As explained in §7.1, one drawback of both word segmentation algorithms is that they produce a deterministic segmentation for each sentence, even though multiple segmentations are possible given the same vocabulary. In contrast, Kudo (2018b) and Provilkov et al. (2020) have proposed methods that enable the model to generate segmentations probabilistically. Instead of selecting the best subword sequence for input x^*, these method stochastically sample a segmentation x' as follows:

$$x' \sim P'(x) \text{ where } P'(x) \propto \begin{cases} P(x) \text{ if } x \in S(x^*) \\ 0 \text{ otherwise} \end{cases}$$

Here we briefly introduce these two methods.

BPE-dropout This method is used together with the BPE algorithm, randomly dropping merge operations with a given probability p while segmenting the input data (Provilkov et al., 2020).

ULM-sample As the ULM algorithm relies on a language model to score segmentation candidates for picking the most likely segmentation, Kudo (2018b) propose to sample from these segmentation candidates based on their language model scores.

7.2.3 Subword Regularization (SR)

Subword regularization (Kudo, 2018b) is a method that incorporates probabilistic segmentation at training time to improve the robustness of models to different segmentations. The idea is conceptually simple: at training time sample different segmentations x' for each input sentence x^*. Previous works (Kudo, 2018b; Provilkov et al., 2020) have demonstrated that subword regularization using both BPE-dropout and ULM-sampling are effective at improving machine translation accuracy, particularly in cross-domain transfer settings where the model is tested on a different domain than the one on which it is trained.

7.3 Subword Regularization for Cross-lingual Transfer

While sub-optimal word segmentation is a challenge in monolingual models, it is an even bigger challenge for multilingual pretrained models. These models train a shared subword segmentation model jointly on data from many languages, but the segmentation can nonetheless be different across languages, stemming from two main issues. First, the granularity of segmentation differs among languages, where the segmentation model tends to *over-segment* low-resource languages that do not have enough representation in the joint training data (Ács, 2019). Fig. 7.2 shows the distribution of words from lan-

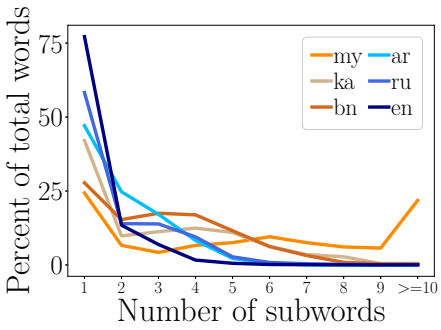

Figure 7.2: Percentage of words with different number of segments from different languages.

guages from different language families based on the number of subwords they are split into.[1] We can see that the majority of English words are not segmented at all, while many languages only have less than half of the words unsegmented. Notably, even though Burmese (my) is a language with little inflectional morphology, almost a quarter of the words are segmented into more than nine subwords. Second, the segmentation might still be *inconsistent* between different languages even if the granularity is similar, as explained in Tab. 7.1. For example, neither the English word "excitement" nor the same word in French "excita/tion" are overly segmented, but segmenting the English word into "excite/ment" would allow the model to learn a better cross-lingual alignment.

Despite these issues, few methods have tried to address this subword segmentation problem for multilingual pretrained models. Chau et al. (2020) propose to adapt a pretrained multilingual model to a new language by augmenting the vocabulary with a new subword vocabulary learned on the target language, but this method might not help for languages other than the target language it adapts to. Chung et al. (2020) propose to separately construct a subword segmentation model for each cluster of related languages for *pretraining* the multilingual representations. However, directly modifying the word segmentation requires retraining large pretrained models, which is computationally prohibitive in most cases.

In this paper, we instead propose a more efficient approach of using probabilistic segmentation during *fine-tuning* on labeled data of a downstream task. As mismatch in segmentation is one of the factors harming cross-lingual transfer, we expect a model that becomes more robust to different varieties of segmentation in one language will be more accommodating to differing segmentations in other languages during inference. Despite the simplicity of this method it is, as far as we are aware, unattested in the literature, and we verify in §7.5.3 that it significantly improves the cross-lingual transfer performance of multilingual pretrained models.

[1] We use Pan et al. (2017a)'s named entity recognition test data with mBERT's tokenizer.

7.4 Multi-view Subword Regularization

Previous attempts at SR have mainly applied it to models trained *from scratch* for tasks such as MT. However, the situation is somewhat different when fine-tuning pre-trained representations, in which case the original pre-trained models are generally not trained on sampled segmentations. This discrepancy between the segmentation of the English labeled data and the segmentation of English monolingual data during pretraining might hurt the ability of the model to take full advantage of the parameters learned during the pretraining stage. To reduce this pretraining–fine-tuning discrepancy, we propose Multi-view Subword Regularization (MVR), a method for learning from multiple segmented versions of the same data and enforcing the consistency of predictions over different segmentations.

Given the input $\widehat{x}_i$ tokenized with the deterministic segmentation such as BPE, and x'_i, the same input tokenized with the corresponding probabilistic segmentation algorithm such as BPE-dropout, the objective for MVR has three components

$$J(\theta) = \sum_{i=1}^{n} \Big[-\frac{1}{2} \underbrace{\log p_\theta(y_i|\widehat{x}_i)}_{\text{Det. Seg CrossEnt}} - \frac{1}{2} \underbrace{\log p_\theta(y_i|x'_i)}_{\text{Prob. Seg CrossEnt}} + \lambda \underbrace{D(p_\theta(y_i|\widehat{x}_i) \,||\, p_\theta(y_i|x'_i))}_{\text{Consistency loss}} \Big] \tag{7.1}$$

1. A cross-entropy loss using the standard deterministic segmentation. This loss acts on data whose segmentation is consistent with the segmentation seen during pretraining. It thus maximizes the benefit of pretrained representations.
2. A cross entropy loss using probabilistic segmentation. It allows the model to learn from different possible segmentations of the same input.
3. A distance term $D(\cdot||\cdot)$ between the model prediction distributions over the two different versions of the input. We use KL divergence as the distance metric and a hyperparameter λ to balance the supervised cross-entropy losses and the consistency loss. Minimizing the distance between the two distributions enforces the model to make consistent predictions under different input segmentations, making it robust to sub-optimal segmentation of multilingual data.[2]

Flattening the prediction The benefit of consistency regularization might be limited if the model prediction becomes overly confident on certain classes, especially when the number of output classes is large. Inspired by a similar technique in knowledge distillation (Hinton et al., 2014), we can use a softmax temperature τ to flatten the prediction distribution when calculating the consistency loss. Specifically, the distance loss between two prediction distributions in Eq. 7.1 can be written as $D(p_\theta^{\text{flat}}(y_i|\widehat{x}_i)||p_\theta(y_i|x'_i))$, where

$$p_\theta^{\text{flat}}(y_i|\widehat{x}_i) = \frac{\exp(z_y)/\tau}{\sum_{y'} \exp(z_{y'})/\tau} \tag{7.2}$$

and z_y is the logit for output label y_i. Normally τ is set to 1, and a higher τ makes the probability distribution more evenly distributed over all classes. In our experiments, we find that $\tau = 1$ works well for most of the tasks and $\tau = 2$ works slightly better for tasks that have larger output label spaces.

[2] As in semi-supervised learning Clark et al. (2018), we expect our method to also be effective when applied to unlabeled data, e.g. using target language adaptation Pfeiffer et al. (2020b), which we leave for future work.

Efficiency At inference time, we simply use the model prediction based on the input tokenized by deterministic segmentation only. Therefore, our method does not add additional decoding latency. MVR needs about twice the fine-tuning cost compared to the baseline. However, compared to pretraining and inference usage of a model, fine-tuning is generally the least expensive component.

7.5 Experiments

7.5.1 Training and evaluation

We evaluate the multilingual representations using tasks from the XTREME benchmark (Hu et al., 2020), focusing on the zero-shot cross-lingual transfer with English as the source language. We consider sentence classification tasks including XNLI (Conneau et al., 2018) and PAWS-X (Yang et al., 2019), a structured prediction task of multilingual NER (Pan et al., 2017a), and question-answering tasks including XQuAD (Artetxe et al., 2020) and MLQA (Lewis et al., 2020b).

7.5.2 Experiment setup

We evaluate on both the mBERT model which utilizes BPE to tokenize the inputs, and the XLM-R models which uses ULM segmentation. To replicate the baseline, we follow the hyperparameters provided in the XTREME codebase[3]. Models are fine-tuned on English training data and zero-shot transferred to other languages. We run each experiment with 5 random seeds and record the average results and the standard deviation.

SR We use BPE-dropout (Provilkov et al., 2020) for mBERT and ULM-sample (Kudo, 2018b) for XLM-R models to do probabilistic segmentation of the English labeled data. BPE-dropout sets a dropout probability of $p \in [0, 1]$ for the merge operations, where a higher p corresponds to stronger regularization. ULM-sample utilizes a sampling temperature $\alpha \in [0, 1]$ to scale the scores for segmentation candidates, and a lower α leads to stronger regularization. We select the p and α values based on the model performance on the English dev set of the NER task and simply use the same values across all other tasks. We set $p = 0.1$ for BPE-dropout and $\alpha = 0.6$ for ULM-sample.

MVR We select the hyperparameters for MVR using the English dev set performance on the NER task. MVR works slightly better by using stronger regularization than SR, likely because using inputs deterministically segmented by the standard algorithm can balance the negative impact of bad tokenization by sampling from a more diverse set of segmentation candidates. We use $\lambda = 0.2, p = 0.2$ for mBERT and $\lambda = 0.6, \alpha = 0.2$ for XLM-R. We use prediction temperature $\tau = 2$ for the question-answering tasks XQuAD and MLQA for the XLMR mdoels, and simply use $\tau = 1$ for all other tasks.

[3] https://github.com/google-research/xtreme

7.5.3 Main results

Model	Method	Avg.	XNLI	PAWS-X	NER	XQuAD	MLQA
Metrics			Acc.	Acc.	F1	F1/EM	F1/EM
mBERT	Hu et al. (2020)	67.1	65.4	81.9	62.2	64.5 / 49.4	61.4 / 44.2
	Baseline (ours)	67.3	66.5±0.4	83.1±0.4	61.5±0.7	64.7±0.2 / 49.8±0.4	60.9±0.4 / 43.8±0.5
	SR	68.0	66.4±0.2	85.0±0.3	62.2±0.6	64.7±0.3 / 50.0±0.3	61.5±0.3 / 44.4±0.3
	MVR	**68.8**	**67.2**±0.3	**85.6**±0.3	**62.7**±0.4	**66.3**±0.2 / **51.7**±0.2	**62.2**±0.2 / **45.3**±0.1
XLM-R base	Baseline (ours)	71.1	74.4±0.2	84.3±0.7	60.6±0.6	70.9±0.3 / 54.9±0.5	65.5±0.3 / 47.7±0.2
	SR	71.4	74.4±0.7	85.5±0.5	61.0±0.6	70.9±0.3 / 55.7±0.2	65.4±0.1 / 47.5±0.1
	MVR	**72.3**	**75.3**±0.3	**86.3**±0.6	**61.8**±0.3	**71.6**±0.5 / **56.5**±0.4	**66.4**±0.5 / **48.5**±0.4
XLM-R large	Hu et al. (2020)	75.8	79.2	86.4	65.4	76.6 / 60.8	71.6 / 53.2
	Baseline (ours)	76.1	80.3±0.4	86.9±0.5	63.6±0.3	77.0±0.2 / 61.7±0.3	**72.8**±0.2 / **54.5**±0.1
	SR	76.5	80.1±0.5	87.3±0.4	65.5±0.6	77.2±0.2 / 62.0±0.2	72.5±0.1 / 54.0±0.2
	MVR	**77.2**	**81.3**±0.1	**88.2**±0.2	**66.0**±0.7	**77.6**±0.2 / **62.5**±0.4	**72.8**±0.2 / **54.5**±0.1

Table 7.2: Average performance and standard deviation of different methods for mBERT, XLM-R base and XLM-R large models. SR is especially effective for mBERT. MVR leads to significant further improvements across all models and tasks.

We compare performance of SR, MVR and the baseline for all models in Tab. 7.2, focusing on the average performance on all languages for each task. Our baseline numbers match or exceed the benchmark results in Hu et al. (2020) for both mBERT and XLM-R large (Hu et al. (2020) do not include results for XLM-R base) on almost all tasks.

Applying SR on English significantly improves other languages SR is surprisingly effective for mBERT—it is comparable to the baseline on XNLI and significantly improves over the baseline for the rest of the four tasks. However, the gains are less consistent for XLM-R models. For both XLM-R base and large, SR leads to improvements on the NER task and the PAWS-X classification task, but is mostly comparable to the baseline for the rest of the three tasks. SR performs better for mBERT likely because the vocabulary of mBERT is more imbalanced than that of XLM-R; it thus benefits more from the regularization methods. mBERT relies on BPE, which could be worse than ULM at tokenizing subwords into morphologically meaningful units (Bostrom and Durrett, 2020). Furthermore, mBERT has only 100K words in the vocabulary while XLM-R has a much larger vocabulary of 250K.

MVR consistently improves over SR For mBERT, it leads to improvements of over 1 to 2 points over the baseline for all tasks. It is also very effective for the XLM-R models. For both the XLM-R base and the stronger XLM-R large models, MVR improves over 1 point over the baseline on the NER task and the two classification tasks. On the question-answering tasks, MVR delivers strong improvements for the XLM-R base model while the improvements on the XLM-R large model is slightly smaller. It has

Method Metrics	Avg.	XNLI Acc.	PAWS-X Acc.	NER F1	XQuAD F1/EM	MLQA F1/EM
MVR	68.8	67.2±0.3	85.6±0.3	62.7±0.4	66.3±0.2 / 51.7±0.2	62.2±0.2 / 45.3±0.1
– Det. Seg CrossEnt	52.8	66.7±0.5	85.5±0.2	62.4±0.7	25.0±8.2 / 15.6±6.7	24.3±8.6 / 13.1±6.4
– Prob. Seg CrossEnt	67.7	66.7±0.6	85.0±0.3	62.3±0.7	64.0±0.3 / 48.7±0.4	60.3±0.4 / 43.1±0.3
– consistency loss	68.2	66.5±0.7	85.3±0.3	62.3±0.6	65.2±0.2 / 50.2±0.4	61.7±0.1 / 44.5±0.2

Table 7.3: Effect of removing each loss component on mBERT.

around 0.5 point improvement on XQuAD and has the same performance on MLQA. MVR leads to more improvements on XQuAD, probably because it has a more diverse set of languages that potentially have more sub-optimal subword segmentation. The consistent gains on both mBERT and XLM-R show that MVR is a general and flexible method for a variety of pretrained multilingual models based on different segmentation methods.

7.5.4 Effect of each loss component

In this section, we verify the effectiveness of the three loss components in MVR by removing each of them from the objective. The ablation results on mBERT for all tasks are listed in Tab. 7.3. Removing any of the three loss components hurts the model performance by about the same amount for most of the tasks. For the question answering tasks, however, removing the cross-entropy loss on the deterministically segmented inputs reduces the model performance by almost half. This is likely because under this setting, the model only learns to locate exact spans for inputs tokenized by BPE-dropout, while we use the standard BPE to segment the inputs at test time.

7.6 Analysis

In this section, we perform several analyses to better understand the behavior and root causes of the accuracy gains realized by our method. The paper corresponding to this chapter also includes some additional analysis of the improvements over languages with Latin vs. non-Latin scripts, which will be included in the final thesis document.

7.6.1 Effect on over-segmentation

In this section, we analyze the effect of our methods on languages and words with different subword segmentation granularity. We focus on the NER task because it contains a diverse set of over 40 languages. We calculate the average number of subword pieces in a language, and plot the gains over the baseline for these languages with respect to their average subwords in Fig. 7.3. To visualize the relationship between the two values, we also fit a trend line and record its coefficient for each method in the legend. We consider three methods for mBERT: SR, MVR without consistency loss, and the full MVR.

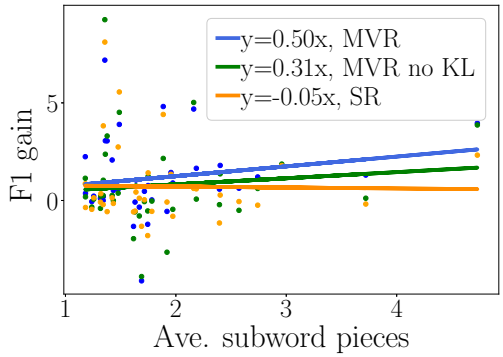

Figure 7.3: mBERT gains over the NER baseline by average word pieces of a language. MVR tends to benefit over-segmented languages more.

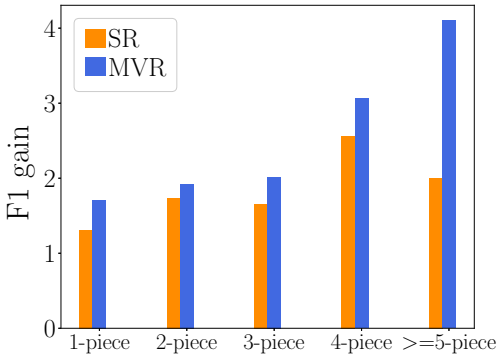

Figure 7.4: XLM-R large gains over the NER baseline for words with increasing number of subword pieces.

The trend line for MVR has a positive coefficient, indicating that it improves more on languages that are more overly segmented. Removing the consistency loss tends to hurt more for these languages. SR, on the other hand, does not tend to favor languages with more subword segmentation.

Next, we bucket all the words together based on how many subwords they are segmented into, and compare the performance of our methods for each word bucket. We use the XLM-R model and plot the results in Fig. 7.4. SR brings slightly more improvements on average for words that are split into 4 or more pieces for the large model. MVR outperforms SR for all categories, especially for difficult words that are segmented into 5 or more subwords.

Gains on Latin vs. non-Latin script In addition, it is notable that we fine-tune the model using labeled data from English, a Latin script language, while the non-Latin scripted languages might have larger segmentation and vocabulary discrepancies from English. We thus also plot the score improvements of both SR and MVR over the baseline for languages with and without Latin script in Fig. 7.5. We use a lighter shade to represent improvements for Latin-script languages and a darker shade for languages with non-Latin scripts. Across all the tasks, both SR and MVR generally have larger improvements on languages with non-Latin script. MVR, which is represented by blue shades, generally outperforms SR for both the Latin and non-Latin scripted languages across all models. While SR sometimes underperforms the baseline on Latin scripted languages, especially for XLM-R models, MVR delivers consistent improvements over the baseline across both types of languages.

7.6.2 Effect of consistency loss

One of the novel components of MVR is the consistency loss between two different segmentations of the input. In this section we analyze two hypotheses about the source of benefit provided thereby.

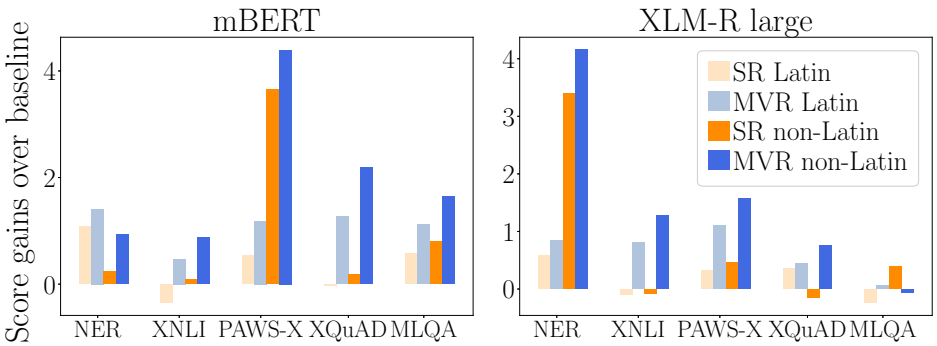

Figure 7.5: Gains over the baseline for languages with Latin vs. non-Latin script. Both SR and MVR improve more for non-Latin languages.

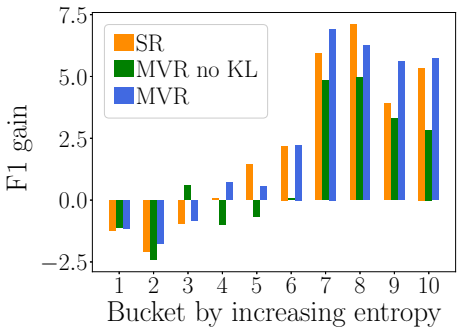

Figure 7.6: mBERT improvements over the NER baseline for examples with different entropy. Consistency loss helps examples with higher entropy more.

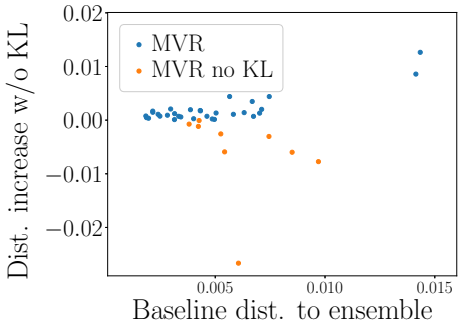

Figure 7.7: Increase in distance to the ensemble distribution by removing the consistency loss on the NER task. The languages are labeled based on the method with closer distance to the ensemble distribution. The consistency loss shifts model prediction closer to the ensemble distribution.

Label smoothing The first hypothesis is that the consistency loss may be able to mitigate overconfident predictions by calibrating the two output distributions against each other. This effect is similar to label smoothing (Szegedy et al., 2015; Yuan et al., 2020), which softens the one-hot target label by adding a loss of uniform distribution over all class labels and has proven helpful across a wide variety of models. To measure this, we plot the F1 improvement on the NER task for examples categorized by increasing predictive entropy in Fig. 7.6. MVR leads to more improvements on examples with higher entropy, or those that the model is more uncertain about, indicating that MVR is indeed helping the model improve on examples where it is not confident.

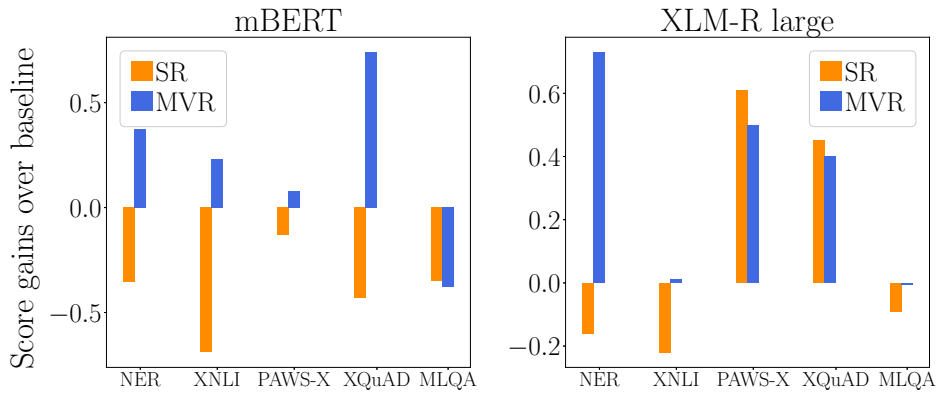

Figure 7.8: Gains of MVR and SR for English. While SR harms the performance on English, MVR generally improves it.

Ensemble effect The second hypothesis is that the consistency loss could regularize the model to be closer to the ensemble of models trained on standard deterministically segmented inputs and probabilistically segmented inputs. To verify this hypothesis, we first calculate the ensembled prediction probability of the baseline and the SR models for each language. Then we compare the KL divergence between this ensemble distribution and MVR with or without the consistency loss. In Fig. 7.7, we plot this KL divergence difference between the MVR without consistency loss and the full MVR for each language in NER. For most of the languages, the full MVR has lower KL divergence with the ensemble distribution, which indicates that the consistency loss trains the model to be closer to the ensemble of two inputs.

7.6.3 MVR also improves English

Although SR improves the model performance averaged over all languages, surprisingly it can hurt the performance on English, the language we use for fine-tuning. Fig. 7.8 shows the improvement on English over the baseline for both SR and MVR, and notably English performance decreases for all tasks on mBERT. MVR, on the other hand, generally brings improvements for English across both mBERT and XLM-R large models. This is likely because MVR also utilizes English inputs with standard segmentation, the method used at pretraining time, which allows it to take full advantage of the information encoded during pretraining.

7.7 Related work

Several works propose to optimize subword-sensitive word encoding methods for pretrained language models. Ma et al. (2020) uses convolutional neural networks (Kim, 2014) on characters to calculate word representations. Zhang and Li (2020) propose to add phrases into the vocabulary for Chinese pretrained language models. However, they focus on improving the vocabulary of pretrained representations of a

single language, and they require modification to the model pretraining stage. Chung et al. (2020) propose to cluster related languages together and run subword vocabulary construction on each language cluster when constructing vocabularies for mBERT. Their method is also applied at the pretraining stage and could be combined with our method for potential additional improvements.

Our method is also related to prior work that optimize word representations for NMT and language modeling. Character level embeddings have been utilized instead of subword segmentation for NMT (Cherry et al., 2018a; Lee et al., 2017; Ataman and Federico, 2018) and language modeling (Kim et al., 2016; Józefowicz et al., 2016). Wang et al. (2019b) propose a multilingual word embedding method for NMT that relies on character n-gram embedding and a latent semantic embedding shared between different languages. Ataman and Federico (2018) show that character n-gram based embedding performs better than BPE for morphologically rich languages. He et al. (2020) propose to learn the optimal segmentation given a subword vocabulary for NMT.

Our method is inspired by semi-supervised learning methods that enforce model consistency on unlabeled data. Several self-training methods utilize unlabeled examples to minimize the distance between the model predictions based on the unlabeled example and a noised version of the same input (Miyato et al., 2017b,a; Xu and Yang, 2017; Clark et al., 2018; Xie et al., 2018). Xu and Yang (2017) use knowledge distillation on unlabeled data to adapt models to a new language. Clark et al. (2018) propose to mask out different parts of the unlabeled input and encourage the model to make consistent prediction given these different inputs. These methods all focus on semi-supervised learning, while our method regulates model consistency to mitigate the subword segmentation discrepancy between different languages.

7.8 Conclusion

The results in this chapter demonstrate that standard deterministic subword segmentation is sub-optimal for multilingual pretrained representations. Even incorporating simple methods for subword regularization such as BPE-dropout at fine-tuning can improve the cross-lingual transfer of pretrained models, and our proposed Multi-view Subword Regularization method further shows consistent and strong improvements over a variety of tasks for models built upon different subword segmentation algorithms. Going forward, we suggest that some variety of subword regularization, MVR or otherwise, should be a standard component of the fine-tuning of pre-trained representations that use subword segmentation.

Part III

Adaptation to language variations

Chapter 8

Efficient Adapter Ensembling for Low-resource Language Varieties

Training a big model using multilingual training is computationally expensive. Therefore, it is important to investigate how we can adapt pretrained multilingual models efficiently to individual language varieties while minimizing the additional parameters required. Adapters are light-weight modules that allow parameter-efficient fine-tuning of pretrained models. Specialized language and task adapters have recently been proposed to facilitate cross-lingual transfer of multilingual pretrained models (Pfeiffer et al., 2020b). In this chapter, we aim to improve the robustness of language adapters to uncovered languages without training new adapters.

8.1 Introduction

Massively multilingual pretrained models (Devlin et al., 2019; Huang et al., 2019; Conneau and Lample, 2019; Conneau et al., 2019) combined with cross-lingual transfer now define the state of the art on a variety of NLP tasks (Hu et al., 2020). Within this paradigm, multilingual pretrained models are fine-tuned on annotated data of a task in a high-resource language, and transferred to other languages. Several recent works propose parameter-efficient fine-tuning methods that insert small *adapter* modules between the layers of pretrained models (Rebuffi et al., 2017; Houlsby et al., 2019). In this line of work, the pretrained model is usually frozen while only the adapters are fine-tuned for a downstream task, which is conducive to both improving the model's learning ability and compactness with respect to storage on disk or in memory. The adapters can be applied to the cross-lingual transfer setting by training separate language and task adapters Pfeiffer et al. (2020b); Üstün et al. (2020). Specifically, Pfeiffer et al. (2020b) propose to perform zero-shot transfer by first training language-level adapters on monolingual data in different languages and then a task adapter on annotated data in the source language.

One drawback of this framework is that a separate language adapter is required for each target language, which is problematic in cases where the data to train these adapters cannot be easily obtained, such as for languages with diverse regional or demographic variations. In fact, certain language varieties

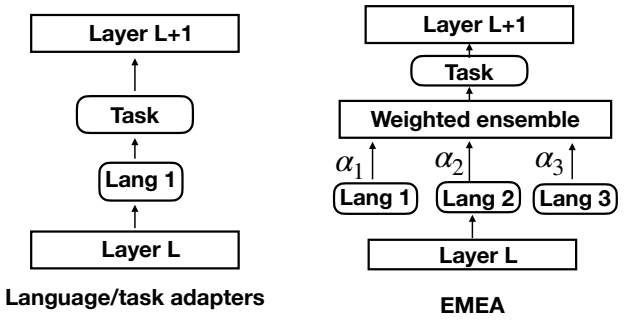

Figure 8.1: Comparison of the standard cross-lingual adapter and our method of entropy minimized ensembling of adapters (EMEA), which combines multiple language adapters to improve robustness to new language varieties at *test time*.

are not included in the standard language identification tools, which makes it challenging to reliably obtain even unlabeled data (Salameh et al., 2018; Caswell et al., 2020a; Demszky et al., 2021). To give just one example, the Nordic languages and dialects form a dialect continuum where the total number of language varieties is difficult to estimate, and language varieties constantly emerge in culturally and linguistically diverse areas (Svendsen and Røyneland, 2008; Røyneland and Jensen, 2020). Although highly related, these language varieties have many systematic differences, which need to be addressed by NLP systems that equitably serve all speakers (Kumar et al., 2021). One potential mitigation strategy is directly using an adapter trained on another similar language variety, but we find this sub-optimal in experiments (§ 8.4).

Instead, we propose two methods to combine existing language adapters to *adapt the model to new language varieties at test time* without any training data. First, we find that simply ensembling multiple related language adapters can significantly improve the fine-tuned model, compared with using individual language adapters. Second, we propose Entropy Minimized Ensemble of Adapters (EMEA; Fig. 9.3), which adapts the ensemble weight of the language adapters for each test instance by minimizing the ensembled model's prediction uncertainty. Our experiments show that EMEA further improves over vanilla ensembling for three groups of uncovered language varieties on both the named entity recognition and part-of-speech tagging tasks.

8.2 Adapters for Cross-lingual Transfer

To facilitate our discussion, we briefly summarize the MAD-X framework (Pfeiffer et al., 2020b) for zero-shot cross-lingual transfer and identify its shortcomings. The goal of MAD-X is to fine-tune a multilingual pretrained model $\mathcal{M}$ to m downstream tasks $T_1, T_2, ..., T_m$, each of which could be in n languages $L_1, L_2, ..., L_n$. To this end, MAD-X relies on language and task adapters, which are light-weight functions inserted in the Transformer layers in $\mathcal{M}$—usually a feed-forward down-projection followed by an up-projection. Specifically, let h be the output of an intermediate layer in $\mathcal{M}$, then $\mathcal{L}_j(h)$ is the transformation that projects h into the embedding space for language L_j, and $\mathcal{T}_i(\mathcal{L}_j(h))$ is the transformation

that projects $\mathcal{L}_j(h)$ into the embedding space for task T_i.

MAD-X trains the adapters $\mathcal{T}_i(\cdot)$ and $\mathcal{L}_j(\cdot)$ in two steps. First, for each language L_j, its adapter $\mathcal{L}_j$ is inserted into $\mathcal{M}$ to replace the output of each layer h with $\mathcal{L}_j(h)$. The resulting model, which we denote as $\mathcal{L}_j \circ \mathcal{M}$, is trained on unlabeled data in L_j using an unsupervised objective such as masked language modeling (MLM; Devlin et al., 2019). Second, for each task T_i, its adapter $\mathcal{T}_i$ is inserted on top of a *src* language adapter $\mathcal{L}_{\text{src}}$. The resulting model $\mathcal{T}_i \circ \mathcal{L}_{\text{src}} \circ \mathcal{M}$ is trained on the downstream task T_i in language L_{src}. After these two steps, $\mathcal{T}_i \circ \mathcal{L}_j \circ \mathcal{M}$ can be used to perform zero-shot cross-lingual transfer for any task T_i and language L_j.

Shortcomings This approach requires a separate adapter for each language one wishes to support. The online database AdapterHub[1] aims to improve the efficiency and reuse of trained language and task adapters (Pfeiffer et al., 2020a) but currently supports only about 50 languages, and hence most languages are not covered. More importantly, as mentioned in the introduction, certain languages have diverse regional varieties and difficulty of reliably obtaining data for them makes adapter-based approaches especially brittle in these cases. In the following § 8.3, we propose strategies to improve the robustness of language adapters to uncovered languages without training new adapters.

8.3 Generalizing Language Adapters to Related Languages

We consider the setting where we have a multilingual pretrained model $\mathcal{M}$ as well as the pretrained task adapters $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_m$ and language adapters $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_n$. We want to use $\mathcal{M}$ and the existing adapters to support a new language L_{new}, which is not in $\{L_1, L_2, ..., L_n\}$ on a given task T without training a new adapter for $\mathcal{L}_{\text{new}}$.

Related Language Adapters One potential solution is to find the most related language $L_{\text{rel}} \in \{L_1, L_2, ..., L_n\}$ and then use $\mathcal{T} \circ \mathcal{L}_{\text{rel}} \circ \mathcal{M}$ to do inference in L_{new}. However, this has two disadvantages. First, the task adapter $\mathcal{T}$ is only trained in the setting of $\mathcal{T} \circ \mathcal{L}_{\text{src}} \circ \mathcal{M}$, so it might not generalize well to the test time setting of $\mathcal{T} \circ \mathcal{L}_{\text{rel}} \circ \mathcal{M}$ (as shown in § 8.4.1). Second, while the pretrained model $\mathcal{M}$ may be relatively robust against distribution shifts (Hendrycks et al., 2020), the specialized language adapters might make the model brittle to language variations because they are trained for specific languages. Our experiments in § 8.4.1 show that this solution indeed leads to poor performance.

Adapter Ensembling As a first solution to this problem, we propose an extremely simple strategy of averaging the transformed outputs of multiple language adapters. Specifically, we use both the source language adapter $\mathcal{L}_{\text{src}}$ and adapters from related languages with similar linguistic properties to the new language. Let $\mathcal{R}$ be the set of the source and related language adapters. To do inference on a task T for the new language L_{new}, we transform the output h of each layer in $\mathcal{M}$ with the language adapters as $\mathcal{L}_{\text{avg}}(h) = \frac{1}{R}\sum_{i=1}^{R} \mathcal{L}_i(h)$.

[1] https://adapterhub.ml/

Algorithm 4: Training with EMEA

Input : Uniform weights α^0, weighted adapter output; $L_{\text{wavg}}(h, \alpha^0)$; test data x; number of update steps T

Output: Prediction $\hat{y}$

for *t in 0, 1, ..., T-1* **do**

  ▷ *Calculate entropy*

  $H(x, \alpha) \leftarrow \text{Entropy}(\mathcal{T} \circ L_{\text{wavg}}(h, \alpha^t) \circ \mathcal{M})$

  ▷ *Calculate gradient*

  $g^t = \nabla_\alpha H(x; \alpha^t)$

  ▷ *Update weighting*

  $\alpha^{t+1} \leftarrow \text{Update}(\alpha^t, g^t)$

end

▷ *Calculate final prediction*

$\hat{y} \leftarrow \text{Predict}(\mathcal{T} \circ L_{\text{wavg}}(h, \alpha^T) \circ \mathcal{M})$

Entropy Minimized Ensemble of Adapters While ensembling is a simple and effective strategy to combine multiple potentially beneficial language adapters, the equal weighing of all language adapters could be sub-optimal for L_{new}; different language varieties, or even sentences, could benefit from a different weighting of the pretrained language adapters. To further improve adapter ensembling, we generalize $\mathcal{L}_{\text{avg}}(h)$ into a learnable weighted average:

$$\mathcal{L}_{\text{wavg}}(h) = \sum_{i=1}^{R} \alpha_i \mathcal{L}_i(h)$$

where $\alpha_1, \alpha_2, ..., \alpha_R$ are learnable weights satisfying $\alpha_i \geq 0$ and $\sum_{i=1}^{S} \alpha_i = 1$. Next, we propose Entropy Minimized Ensemble of Adapters (EMEA) method, which learns the adapter weightings *for each sentence* without additional training.

The intuition behind our method is that a good adapter weight α for a test input x should make the model more confident in its prediction for x, that is, it should lead to lower model entropy over the input (Shannon, 1948; Wang et al., 2021a). Specifically for structured prediction tasks, we want to classify each word x_w in a test input x with W words into one of the possible C classes. We consider the entropy: $H(x; \alpha) = -\sum_{w=1}^{W} \sum_{c=1}^{C} P(c|x_w; \alpha) \log P(c|x_w; \alpha)$, where $P(c|x_w; \alpha)$ is the prediction of the model $\mathcal{T} \circ L_{\text{wavg}}(h) \circ \mathcal{M}$. Since $P(c|x_w; \alpha)$ is a function of the ensemble weights α, we can calculate the gradient of α as $g_i = \nabla_{\alpha_i} H(x; \alpha)$.

To minimize the entropy loss, we can simply do gradient descent steps on each α_i using the corresponding gradient g_i by $\alpha_i = \alpha_i - \gamma g_i$, where γ is the learning rate. We can then use the updated α to calculate the final prediction for x. In § 8.4, we find that a single step of gradient update already leads to better performance than simple ensembling. We can additionally perform multiple steps of gradient descent to obtain a better α at the cost of lower inference speed. Alg. 4 shows the pseudo code of our method[2].

[2]Code can be found at https://github.com/cindyxinyiwang/emea

Related	Additional	Test
hi	en,ar	mr,bn,ta,bho
is	en,de	fo,no,da
ru	en	be,uk,bg

Table 8.1: Test language groups and their corresponding language adapters. Adapters from languages in the first two columns are applied to the test languages in the third column.

8.4 Experiments

Data We focus on zero-shot cross-lingual transfer with English as the source language. We conduct experiments on named entity recognition (NER) and part-of-speech tagging (POS). We use the WikiAnn dataset (Pan et al., 2017a) for NER and Universial Treebank 2.0 for POS tagging (Nivre et al., 2018).

Model We use the mBERT (Devlin et al., 2019) model, which shows good performance for low-resource languages on the structured prediction tasks (Pfeiffer et al., 2020b; Hu et al., 2020). We use the English annotated data to train the task adapter. Each experiment is run with 5 different random seeds and we report the average performance.

Languages We focus on three related language groups with rich regional variations. Each group has a language with pretrained adapter available on the AdapterHub (Pfeiffer et al., 2020a), and we test on the languages without adapters. The language with adapter and the corresponding target languages for each group are: 1. Hindi (hi): Marathi (mr), Bengali (bn), Tamil (ta), Bhojpuri (bho); 2. Icelandic (is): Faroese (fo), Norwegian (no), Danish (da); 3. Russian (ru): Bulgarian (bg), Ukrainian (uk), Belorussian (be). We only test on languages with data available for each task. For our methods, we additionally use the English adapter (the src language), and optionally another highly related language if there is one available on the AdapterHub. The adapters used are listed in Tab. 8.1.

Baselines We compare with several competitive baselines: 1) En: the English adapter; 2) Related: the best performing related language adapter; 3) Continual learning (CL): we use the English language adapter and update its parameters using the entropy loss for each test input; 4) Fusion: learn another set of key, value and query parameters in each layer that uses the layer output as query to mix together the output of each adapter (Pfeiffer et al., 2021a). Since we do not use labeled data in the new language, we train the fusion parameters on English labeled data.

Implementation Details We preprocess the data using scripts in XTREME (Hu et al., 2020). We use the best performing adapter configuration in Pfeiffer et al. (2020b). For NER, we train the task adapter for 100 epochs using learning rate of 1e-4. For POS tagging, we train the task adapter for 50 epochs

Task	Method	mr	bn	ta	avg.	fo	no	da	avg.	be	uk	bg	avg.	avg.
NER	En	48.0	54.4	29.6	44.0	57.5	73.3	80.5	70.4	67.1	67.6	71.1	68.6	61.0
	Related	51.7	47.0	30.8	43.1	54.3	72.7	79.3	68.7	66.2	65.8	69.8	67.3	59.7
	CL	48.1	55.2	28.9	44.1	57.5	73.6	80.6	70.6	67.0	67.8	71.0	68.6	61.1
	Fusion	51.7	47.0	30.8	43.1	**58.8**	73.3	80.9	71.0	66.2	65.8	69.8	67.3	60.5
	Ensemble	55.6	55.3	35.8	48.9	57.7	73.8	80.3	70.6	70.7	70.5	74.0	71.7	63.7
	EMEA	**55.9**	**58.3**	**36.3**	**50.1**	**58.8**	**74.2**	**81.1**	**71.3**	**71.3**	**70.7**	**74.3**	**72.1**	**64.5**
	Method	mr	bho	ta	avg.	fo	no	da	avg.	be	uk	bg	avg.	avg.
POS	En	62.6	39.5	53.4	51.8	71.6	**84.6**	87.6	81.1	85.3	81.4	84.6	83.7	72.2
	Related	53.2	**46.9**	47.0	49.0	72.8	82.4	86.9	80.7	84.0	79.5	82.9	82.1	70.6
	CL	62.6	39.6	53.6	51.9	71.7	84.2	87.7	81.2	85.6	81.5	84.7	83.9	72.3
	Fusion	54.5	44.1	51.6	50.1	72.0	80.2	84.4	78.9	83.7	78.6	82.2	81.5	70.1
	Ensemble	61.0	45.3	53.5	53.2	73.8	83.3	87.7	81.6	85.9	**81.6**	84.6	84.0	72.9
	EMEA	**63.2**	45.0	**54.0**	**54.0**	**74.0**	83.6	**87.9**	**81.8**	**86.0**	**81.6**	**84.7**	**84.1**	**73.3**

Table 8.2: F1 of the baselines and our methods for each language group.

with the learning rate of 1e-4. For EMEA, we search over the learning rate γ of $0.1, 1, 10$ on the English validation set and pick $\gamma = 10$ for all experiments.

For Fusion, we use learning rate of 5e-5 which is recommended by (Pfeiffer et al., 2021a). We search over the best learning rate for CL on the performance of English labeled data. We use the learning rate of 2e-5 and do 1 step of gradient update for each batch. For our experiment on training new adapters, we find that training from scratch on `no` and `mr` is not competitive when using very small amount of data. Therefore, we continue training from their related language adapters.

8.4.1 Results

The main results of our method on NER and POS tagging can be found in Tab. 8.2. For most languages simply using the English adapter is better than the best individual related language adapter. This verifies our hypothesis that specialized language adapters might not be robust to language variations. CL leads to slight improvements to some languages but is generally comparable to En. Fusion does not work well for most language varieties, especially the non-Latin script languages, probably because it can overfit to the English labeled data.

Using multiple language adapters brings significant gains Ensembling leads to significant gains for the non-Latin language group. It also brings improvements or is comparable to the best baseline on other languages. EMEA delivers further improvements across almost all languages, demonstrating the effectiveness of adapting language adapter weights to each test sentence. We hypothesize that the proposed methods improve non-Latin languages more because these are low-performing languages that

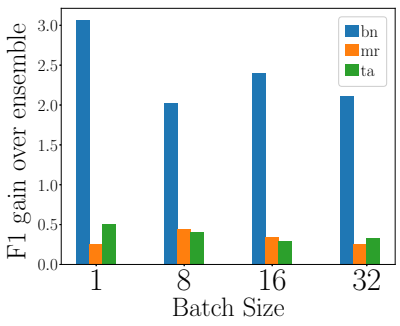

Figure 8.2: Improvements over ensemble with different batch size.

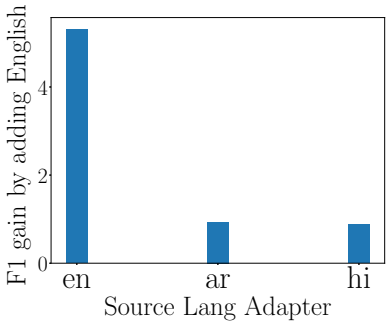

Figure 8.3: Improvements by adding en adapter for different src language adapters.

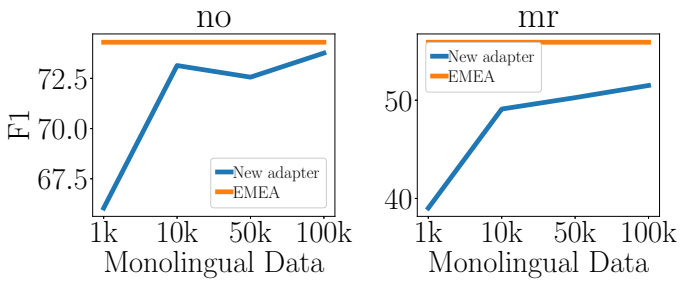

Figure 8.4: Comparison to training adapter on different amount of monolingual data.

the model is more uncertain about.

8.5 Analysis

Effect of test batch size In Fig. 8.2 we plot the result of using different test batch size with EMEA on the NER task. A smaller batch size leads to more fine-grained test time adaptation with a higher computational cost. Fig. 8.2 shows that smaller batch size indeed leads to better performance while using a larger batch size still outperforms the baseline.

Significance of source language adapter We investigate whether the benefit of adding the src language adapter comes from the discrepancy between training and testing of the task adapter. We train different task adapters with language adapters other than English(en), and compare the improvement of adding the en adapter for the ensemble. Fig. 8.3 shows that the en adapter provides the largest benefit when it is used to train the task adapter, which verifies that using different language adapters with the task adapter between training and testing leads to sub-optimal cross-lingual transfer performance.

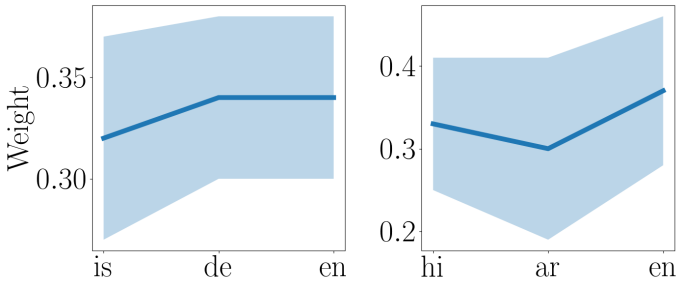

Figure 8.5: Mean and standard deviation of the weight for each adapter for the `is` (*left*) and `hi` (*right*) language groups.

Comparison to training new adapters In order to understand the data requirement of training new language adapters, we also examine the method that trains new adapters using small amount of monolingual data in the target language. We focus on two languages, mr and no, on the NER task, and the results are in Fig. 8.4. Note that this is not a fair comparison since EMEA does not require any training. It takes about 100k monolingual data for no to reach comparable performance with our method, while mr still lags behind EMEA.

Analysis of weights We plot the mean and standard deviation of ensembling weights from EMEA in Fig. 8.5. The En adapter gets the highest weight for both language groups, in line with the results in Tab. 8.2 showing `en` as the best individual adapter. For the `hi` language group, the `ar` adapter tends to have the least benefit, probably because it has a different script from the languages we test on.

8.6 Conclusion

We find that specialized language adapters might not be robust to language variations, and that utilization of multiple existing pretrained language adapters alleviates this issue. We hope our findings can inspire future work on models that are robust and adaptive to language variations.

Chapter 9

Expanding Pretrained Models to Thousands More Languages via Lexicon-based Adaptation

The previous chapter explores methods that adapt pretrained models to language variations or dialects. Here we focus on adapting pretrained multilingual models to another type of low-resource languages – languages that have little or no textual data. The performance of multilingual pretrained models is highly dependent on the availability of monolingual or parallel text present in a target language. To expand possibilities of using NLP technology in these under-represented languages, we systematically study strategies that relax the reliance on conventional language resources through the use of bilingual lexicons, an alternative resource with much better language coverage.[1]

9.1 Introduction

Multilingual pretrained models (Devlin et al., 2019; Conneau and Lample, 2019; Conneau et al., 2019) have become an essential method for cross-lingual transfer on a variety of NLP tasks (Pires et al., 2019; Wu and Dredze, 2019). These models are supposed to benefit under-represented languages that do not have annotated data. However, recent studies have found that the cross-lingual transfer performance of a language is highly contingent on the availability of monolingual data in the language during pretraining Hu et al. (2020). Languages with more monolingual data tend to have better performance while languages not present during pretraining significantly lag behind.

Several works propose methods to adapt the pretrained multilingual models to low-resource languages, but these generally involve continued training using monolingual text from these languages (Wang et al., 2020b; Chau et al., 2020; Pfeiffer et al., 2020b, 2021b). Therefore, the performance of these methods is still constrained by the amount of monolingual or parallel text available, making it difficult for lan-

[1]Code and data are available at: https://github.com/cindyxinyiwang/expand-via-lexicon-based-adaptation.

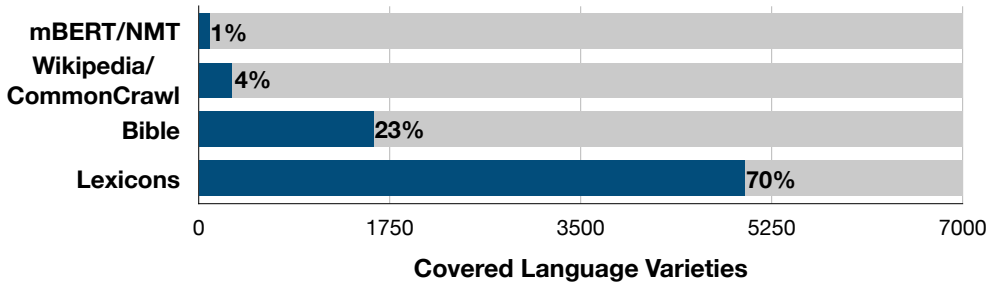

Figure 9.1: The percentage of the world's ≈7,000 languages covered by mBERT, monolingual data sources and lexicons.

guages with little or no textual data to benefit from the progress in pretrained models. Joshi et al. (2020) indeed argue that unsupervised pretraining makes the 'resource-poor poorer'.

Fig. 9.1 plots the language coverage of multilingual BERT (mBERT; Devlin et al., 2019), a widely used pre-trained model, and several commonly used textual data sources.[2] Among the 7,000 languages in the world, mBERT only covers about 1% of the languages while Wikipedia and CommonCrawl, the two most common resources used for pretraining and adaptation, only contain textual data from 4% of the languages (often in quite small quantities, partially because language IDs are difficult to obtain for low-resource languages (Caswell et al., 2020b)). Ebrahimi and Kann (2021) show that continued pretraining of multilingual models on a small amount of Bible data can significantly improve the performance of uncovered languages. Although the Bible has much better language coverage of 23%, its relatively small data size and constrained domain limits its utility (see § 9.6)—and 70% of the world's languages do not even have this resource. The failure of technology to adapt to these situations raises grave concerns regarding the fairness of allocation of any benefit that may be conferred by NLP to speakers of these languages (Joshi et al., 2020; Blasi et al., 2022). On the other hand, linguists have been studying and documenting under-represented languages for years in a variety of formats (Gippert et al., 2006). Among these, bilingual lexicons or word lists are usually one of the first products of language documentation, and thus have much better coverage of the worlds' languages than easily accessible monolingual text, as shown in Fig. 9.1. There are also ongoing efforts to create these word lists for even more languages through methodologies such as "rapid word collection" (Boerger, 2017), which can create an extensive lexicon for a new language in a number of days. As Bird (2020) notes:

> After centuries of colonisation, missionary endeavours, and linguistic fieldwork, all languages have been identified and classified. There is always a wordlist. . . . In short, we do not need to "discover" the language ex nihilo (L1 acquisition) but to leverage the available resources (L2 acquisition).

However, there are few efforts on understanding the best strategy to utilize this valuable resource for adapting pretrained language models. Bilingual lexicons have been used to synthesize bilingual data for learning cross-lingual word embeddings Gouws and Søgaard (2015); Ruder et al. (2019) and task data for NER via word-to-word translation (Mayhew et al., 2017), but both approaches precede the adoption of pre-trained multilingual LMs. Khemchandani et al. (2021) use lexicons to synthesize monolingual data

[2] Statistics taken from Ebrahimi and Kann (2021) and panlex.org.

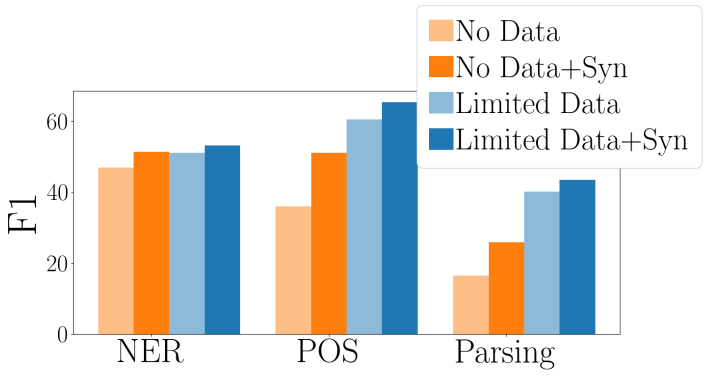

Figure 9.2: Results for baselines and adaptation using synthetic data for both resource settings across three NLP tasks.

for adapting LMs, but their experimentation is limited to several Indian languages and no attempt was made to synthesize downstream task data while Hu et al. (2021) argue that bilingual lexicons may hurt performance.

In this paper, we conduct a systematic study of strategies to leverage this relatively under-studied resource of bilingual lexicons to adapt pretrained multilingual models to languages with little or no monolingual data. Utilizing lexicons from an open-source database, we create synthetic data for both continued pretraining and downstream task fine-tuning via word-to-word translation. Empirical results on 19 under-represented languages on 3 different tasks demonstrate that using synthetic data leads to significant improvements on all tasks (Fig. 9.2), and that the best strategy depends on the availability of monolingual data (§ 9.5, § 9.6). We further investigate methods that improve the quality of the synthetic data through a small amount of parallel data or by model distillation.

9.2 Background

We focus on the cross-lingual transfer setting where the goal is to maximize performance on a downstream task in a target language T. Due to the frequent unavailability of labeled data in the target language, a pretrained multilingual model M is typically fine-tuned on labeled data in the downstream task $\mathcal{D}^S_{label} = \{(x^S_i, y^S_i)\}^N_{i=1}$ in a source language S where x^S_i is a textual input, y^S_i is the label, and N is the number of labeled examples. The fine-tuned model is then directly applied to task data $\mathcal{D}^T_{test} = \{x^T_i, y^T_i\}_i$ in language T at test time.[3] The performance on the target language T can often be improved by further adaptation of the pretrained model.

[3]We additionally examine the few-shot setting where some task data $\mathcal{D}^T_{label}$ in T is available for fine-tuning in § 9.7.

9.2.1 Adaptation with Text

There are two widely adopted paradigms for adapting pretrained models to a target language using monolingual or parallel text.

MLM Continued pretraining on monolingual text $\mathcal{D}^T_{mono} = \{x^T_i\}_i$ in the target language (Howard and Ruder, 2018b; Gururangan et al., 2020) using a masked language model (MLM) objective has proven effective for adapting models to the target language (Pfeiffer et al., 2020b). Notably, Ebrahimi and Kann (2021) show that using as little as several thousand sentences can significantly improve the model's performance on target languages not covered during pretraining.

Trans-Train For target languages with sufficient parallel text with the source language $\mathcal{D}^{ST}_{par} = \{(x^S_i, x^T_i)\}_i$, one can train a machine translation (MT) system that translates data from the source language into the target language. Using such an MT system, we can translate the labeled data in the source language $\mathcal{D}^S_{label}$ into target language data $\widehat{\mathcal{D}}^T_{label} = \{(\widehat{x}^T_i, y^S_i)\}^N_{i=1}$, and fine-tune the pretrained multilingual model on both the source and translated labeled data $\mathcal{D}^S_{label} \cup \widehat{\mathcal{D}}^T_{label}$. This method often brings significant gains to the target language, especially for languages with high-quality MT systems (Hu et al., 2020; Ruder et al., 2021).

9.2.2 Challenges with Low-resource Languages

Both methods above require $\mathcal{D}^T_{mono}$ or $\mathcal{D}^{ST}_{par}$ in target language T, so they cannot be directly extended to languages without this variety of data. Joshi et al. (2020) classified the around 7,000 languages of the world into six groups based on the availability of data in each language. The two groups posing the biggest challenges for NLP are:

"The Left-Behinds," languages with virtually no unlabeled data. We refer to this as the *No-Text* setting.

"The Scraping-Bys," languages with a small amount of monolingual data. We refer to this as the *Few-Text* setting.

These languages make up 85% of languages in the world, yet they do not benefit from the development of pretrained models and adaptation methods due to the lack of monolingual and parallel text. In this paper, we conduct a systematic study of strategies directly targeted at these languages.

9.3 Adapting to Under-represented Languages Using Lexicons

Since the main bottleneck of adapting to under-represented languages is the lack of text, we adopt a data augmentation framework (illustrated in Fig. 9.3) that leverages bilingual lexicons, which are available for a much larger number of languages.

eng $x^S \in \mathcal{D}^S_{mono}$	Anarchism calls for the abolition of the state , which it holds to be undesirable , unnecessary , and harmful .
Pseudo Mono $\widetilde{x}^T \in \widetilde{\mathcal{D}}^T_{mono}$	Anarchism calls għal il abolition ta' il stat , lima hi holds għal tkun undesirable , bla bzonn , u harmful .
eng $x^S \in \mathcal{D}^S_{label}$	I suspect the streets of Baghdad will look as if a war is looming this week .
Pseudo Labeled $\widetilde{x}^T \in \widetilde{\mathcal{D}}^T_{label}$	jien iddubita il streets ta' Bagdad xewqa hares kif jekk a gwerra is looming dan ġimgħa .
Pseudo Labeled $y^S \in \widetilde{\mathcal{D}}^T_{label}$	PRON VERB DET NOUN ADP PROPN AUX VERB SCONJ SCONJ DET NOUN AUX VERB DET NOUN PUNCT
Label Distilled $\widetilde{y}^T \in \widetilde{\mathcal{D}}^T_{distill}$	PRON VERB DET NOUN ADP PROPN NOUN NOUN SCONJ SCONJ DET NOUN AUX VERB DET NOUN PUNCT

Table 9.1: Examples of pseudo monolingual data and pseudo labeled data for POS tagging for Maltese (mlt). Words in red have different labels between the source language and the label distilled data. This is because "xewqa" in Maltese is a noun meaning "desire,will", while the word "will" is not used as a noun in the original English sentence.

9.3.1 Synthesizing Data Using Lexicons

Given a bilingual lexicon $\mathcal{D}^{ST}_{\text{lex}}$ between the source language S and a target language T, we create synthetic sentences $\widetilde{x}^T_i$ in T using sentences x^S_i in S via word-to-word translation, and use this synthetic data in the following adaptation methods.

Pseudo MLM Using monolingual text $\mathcal{D}^S_{mono} = \{x^S_i\}_i$, we generate pseudo monolingual text $\widetilde{\mathcal{D}}^T_{mono} = \{\widetilde{x}^T_i\}_i$ for T by replacing the words in x^S_i with their translation in T based on the lexicon $\mathcal{D}^{ST}_{\text{lex}}$. We keep the words that do not exist in the lexicon unchanged, so the pseudo text $\widetilde{x}^T_i$ can include words in both S and T. We then adapt the pretrained multilingual model on $\widetilde{\mathcal{D}}^T_{mono}$ using the MLM objective. For the Few-Text setting where some gold monolingual data $\mathcal{D}^T_{mono}$ is available, we can train the model jointly on the pseudo and the gold monolingual data $\widetilde{\mathcal{D}}^T_{mono} \cup \mathcal{D}^T_{mono}$.

Pseudo Trans-train Given the source labeled data $\mathcal{D}^S_{label} = \{(x^S_i, y^S_i)\}^N_{i=1}$, for each text example x^S_i we use $\mathcal{D}^{ST}_{\text{lex}}$ to replace the words in x^S_i with its corresponding translation in T, resulting in pseudo labeled data $\widetilde{\mathcal{D}}^T_{label} = \{(\widetilde{x}^T_i, y^S_i)\}^N_{i=1}$. We keep the original word if it does not have an entry in the lexicon. We then fine-tune the model jointly on both pseudo and gold labeled data $\widetilde{\mathcal{D}}^T_{label} \cup \mathcal{D}^S_{label}$.

Since these methods only require bilingual lexicons, we can apply them to both No-Text and Few-Text settings. We can use either of the two methods or the combination of both to adapt the model.

Challenges with Pseudo Data Our synthetic data $\widetilde{\mathcal{D}}^T$ could be very different from the true data $\mathcal{D}^T$ because the lexicons do not cover all words in S or T, and we do not consider morphological or word order differences between T and S.[4] Nonetheless, we find that this approach yields significant improvements in practice (see Tab. 9.3). We also outline two strategies that aim to improve the quality of the synthetic data in the next section.

[4]In fact, we considered more sophisticated methods using morphological analyzers and inflectors, but even models with relatively broad coverage (Anastasopoulos and Neubig, 2019) did not cover many languages we used in experiments.

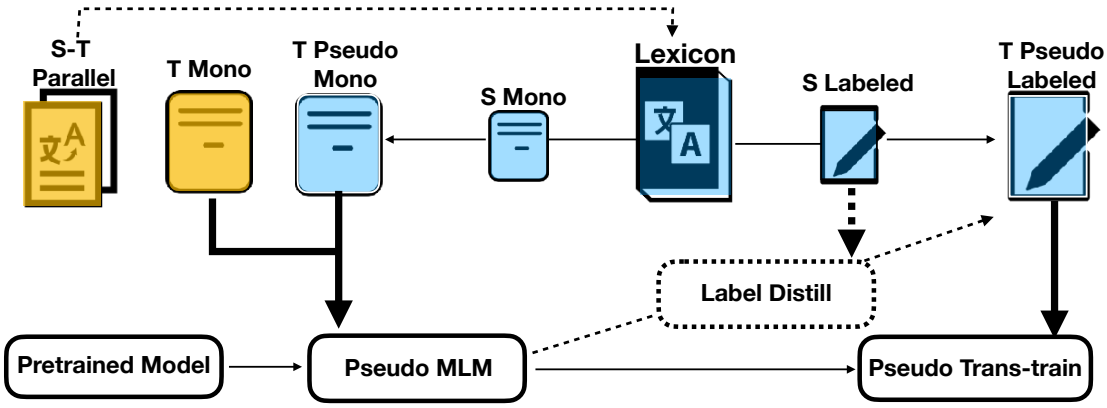

Figure 9.3: Pipelines for synthesizing data for both No-text and Few-text settings and utilizing extra data for the Few-Text setting. Solid lines indicate adaptation methods and dashed lines are synthetic data refinement methods.

9.3.2 Refining the Synthetic Data

Label Distillation The pseudo labeled data $\widetilde{\mathcal{D}}^T_{label} = \{(\widetilde{x}^T_i, y^S_i)\}^N_{i=1}$ is noisy because the synthetic examples $\widetilde{x}^T_i$ could have a different label from the original label y^S_i (see Tab. 9.1). To alleviate this issue, we propose to automatically "correct" the labels of pseudo data using a teacher model. Specifically, we fine-tune the pretrained multilingual model as a teacher model using only $\mathcal{D}^S_{label}$. We use this model to generate the new pseudo labeled data $\widetilde{\mathcal{D}}^T_{distill} = \{(\widetilde{x}^T_i, \widetilde{y}^T_i)\}^N_{i=1}$ by predicting labels $\widetilde{y}^T_i$ for the pseudo task examples $\widetilde{x}^T_i$. We then fine-tune the pretrained model on both the new pseudo labeled data and the source labeled data $\widetilde{\mathcal{D}}^T_{distill} \cup \mathcal{D}^S_{label}$.

Induced Lexicons with Parallel Data For the Few-Text setting, we can leverage the available parallel data $\mathcal{D}^{ST}_{par}$ to further improve the quality of the augmented data. Specifically, we use unsupervised word alignment to extract additional word pairs $\widetilde{\mathcal{D}}^{ST}_{lex}$ from the parallel data, and use the combined lexicon $\widetilde{\mathcal{D}}^{ST}_{lex} \cup \mathcal{D}^{ST}_{lex}$ to synthesize the pseudo data.

9.4 General Experimental Setting

In this section, we outline the tasks and data setting used by all experiments. We will then introduce the adaptation methods and results for the No-Text setting in § 9.5 and the Few-Text setting in § 9.6.

9.4.1 Tasks, Languages and Model

We evaluate on the gold test sets of three different tasks with relatively good coverage of under-represented languages: named entity recognition (NER), part-of-speech (POS) tagging, and dependency parsing (DEP). We use two NER datasets: WikiAnn NER (Pan et al., 2017b; Rahimi et al., 2019) and MasakhaNER (Adelani et al., 2021). We use the Universal Dependency 2.5 (Nivre et al., 2018) dataset for both the POS and

Language	iso	Family	Task	Lex Count
Acehnese	ace	Austronesian	NER	0.5k
Bashkir	bak	Turkic	NER	3.4k
Crimean Turkish	crh	Turkic	NER	4.4k
Hakka Chinese	hak	Sino-Tibetan	NER	8.5k
Igbo	ibo	Niger-Congo	NER	3.6k
Ilokano	ilo	Austronesian	NER	4.0k
Kinyarwanda	kin	Niger-Congo	NER	4.7k
Eastern Mari	mhr	Uralic	NER	21.7k
Maltese	mlt	Afro-Asiatic	All	1.0k
Maori	mri	Austronesian	NER	13.8k
Hausa	hau	Niger-Congo	NER	5.6k
Wolof	wol	Niger-Congo	All	1.9k
Luganda	lug	Niger-Congo	NER	3.5k
Luo	luo		NER	0.7k
Bambara	bam	Mande	POS,Parsing	4.4k
Manx	glv	Indo-European	POS,Parsing	37.6k
Ancient Greek	grc	Indo-European	POS,Parsing	8.0k
Swiss German	gsw	Indo-European	POS,Parsing	2.5k
Erzya	myv	Uralic	POS,Parsing	7.4k

Table 9.2: Languages used for evaluation.

DEP tasks.

We use English as the source language for all experiments. For each dataset, we use the English training data and select the checkpoint with the best performance on the English development set. For MasakhaNER, which does not have English training data, we follow Adelani et al. (2021) and use the CoNLL-2003 English NER training data. We run each fine-tuning experiment with 3 random seeds and report the average performance. For NER and POS tagging, we follow the data processing and fine-tuning hyper-parameters in Hu et al. (2020). We use the Udify (Kondratyuk and Straka, 2019) codebase and configuration for parsing.

Languages For each task, we select languages that have task data but are not covered by the mBERT pretraining data. The languages we use can be found in Tab. 9.2. Most fall under the Few-Text setting (Joshi et al., 2020). We employ the same languages to simulate the No-Text setting as well.

Model We use the multilingual BERT model (mBERT) because it has competitive performance on under-represented languages (Pfeiffer et al., 2020b). We find that our mBERT performance on WikiNER and POS is generally comparable or exceeds the XLM-R large results in Ebrahimi and Kann (2021). We additionally verify our results also hold for XLM-R in § 9.7.

9.4.2 Adaptation Data

Lexicon We extract lexicons between English and each target language from the PanLex database.[5] The number of lexicon entries varies from about 0.5k to 30k, and most of the lexicons have around 5k entries. The lexicon statistics for each language can be found in Tab. 9.2.

Pseudo Monolingual Data English Wikipedia articles are used to synthesize monolingual data. We first tokenize the English articles using Stanza (Qi et al., 2020) and keep the first 200k sentences. To create pseudo monolingual data for a given target language, we replace each English word with its translation if the word exists in the bilingual lexicon. We randomly sample a target word if the English word has multiple possible translations because it is difficult to estimate translation probabilities due to lack of target text.

Pseudo Labeled Data Using the English training data for each task, we simply replace each English word in the labeled training data with its corresponding translation and retain its original label. For the sake of simplicity, we only use lexicon entries with a single word.

9.5 No-Text Setting

We analyze the results of the following adaptation methods for the setting where we do not have any monolingual data.

Pseudo MLM The mBERT model is trained on the pseudo monolingual data using the MLM objective. We train the model for 5k steps for the NER tasks and 10k steps for the POS tagging and Parsing tasks.

Pseudo Trans-train We fine-tune mBERT or the model adapted with Pseudo MLM for a downstream task on the concatenation of both the English labeled data and the pseudo labeled data.

Label Distillation We use the model adapted with Pseudo MLM as the teacher model to generate new labels for the pseudo labeled data, which we use jointly with the English labeled data to fine-tune the final model.

[5] https://panlex.org/snapshot/

	Method	Lexicon	WikiNER	Δ	MasakhaNER	Δ	POS	Δ	Parsing	Δ	Avg.	Δ
No-Text	mBERT	-	47.6	-	46.1	-	36.1	-	16.5	-	36.5	-
	Pseudo Trans-train	PanLex	49.8	2.2	54.4	8.3	**51.1**	15.0	25.9	9.4	**45.2***	8.7
	Pseudo MLM	PanLex	49.8	2.2	52.6	6.5	48.9	12.8	25.2	8.7	44.1*	7.6
	Both	PanLex	48.5	0.9	**54.6**	8.5	48.7	12.6	25.9	9.4	44.4*	7.9
	Both+Label Distillation	PanLex	**50.6**	2.1	53.5	-1.1	50.3	1.6	**26.0**	0.1	45.1*	0.7
Few-Text	Gold MLM	-	49.5	-	53.6	-	60.6	-	40.2	-	50.9	-
	Pseudo Trans-train	PanLex	50.2	0.7	59.4	5.8	59.3	-1.3	37.0	-3.2	51.4	0.5
	Pseudo MLM	PanLex	50.7	1.2	57.4	3.8	65.4	4.8	**43.5**	3.3	54.2*	3.3
		PanLex+Induced	52.2	1.5	58.5	0.9	64.7	-0.7	41.5	-2.0	54.2*	0.0
	Both	PanLex	50.1	0.6	59.2	5.6	60.7	0.1	38.3	-1.9	52.0*	1.1
		PanLex+Induced	52.6	2.5	**61.1**	1.9	59.5	-1.2	35.3	-3.0	52.0†	0.0
	Both+Label Distillation	PanLex	51.7	1.6	58.4	-0.8	**66.2**	5.5	41.9	3.6	54.5*	2.5
		PanLex+Induced	**53.2**	1.5	59.4	1.0	65.8	-0.4	40.7	-1.2	**54.7***	0.2

Table 9.3: Average F1 score for languages in each task. We record F1 of the LAS for Parsing. We compare three adaptation methods (Δ indicates gains over baselines): Pseudo Trans-train, Pseudo MLM, and Both. We also examine two data refinement methods: Label Distillation (Δ is gains over Both) and PanLex+Induced (Δ is gains over PanLex). **Bold** is the best result for each dataset, and underline indicates the best improvements among the three adaptation methods over the baselines. We test the significance of the average gains over the baselines in the last column using paired bootstrap resampling. * indicates significant gains with $p < 0.001$ and † indicates significant gains with $p < 0.05$.

9.5.1 Results

The average performance of different adaptation methods averaged across all languages in each task can be found in Tab. 9.3.

Pseudo Trans-train is the best method for No-Text. Pseudo MLM and Pseudo Trans-train can both bring significant improvements over the mBERT baseline for all tasks. Pseudo Trans-train leads to the best aggregated result across all tasks, and it is also the best method or very close to the best method for each task. Adding Pseudo Trans-train on top of Pseudo MLM does not add much improvement. Label Distillation generally leads to better performance, but overall it is comparable to only using Pseudo Trans-train.

9.6 Few-Text Setting

We test same adaptation methods introduced in § 9.5 for the Few-Text setting where we have a small amount of gold data. First we introduce the additional data and adaptation methods for this setting.

9.6.1 Gold Data

Gold Monolingual Data We use the JHU Bible Corpus (McCarthy et al., 2020) as the monolingual data. Following the setup in Ebrahimi and Kann (2021), we use the verses from the New Testament, which contain 5000 to 8000 sentences for each target language.

Gold Parallel Data We can use the parallel data between English and the target languages from the Bible to extract additional word pairs. We use an existing unsupervised word alignment tool, eflomal (**?**), to generate word alignments for each sentence in the parallel Bible data. To create high quality lexicon entries, we only keep the word pairs that are aligned more than once, resulting in about 2k extra word pairs for each language. We then augment the PanLex lexicons with the induced lexicon entries.

9.6.2 Adaptation Methods

Gold MLM The mBERT model is trained on the gold monolingual Bible data in the target language using the MLM objective. Following the setting in Ebrahimi and Kann (2021), we train for 40 epochs for the NER task, and 80 epochs for the POS and Parsing tasks.

Pseudo MLM We conduct MLM training on both the Bible monolingual data and the pseudo monolingual data in the target language. The Bible data is up-sampled to match the size of the pseudo monolingual data. We train the model for 5k steps for the NER task and 10k steps for the POS tagging and Parsing tasks.

9.6.3 Results

The average performance in each task for Few-Text can be found in Tab. 9.3.

Pseudo MLM is the competitive strategy for Few-Text. Unlike the No-Text setting, Pseudo Trans-train only marginally improves or even decreases the performance for three out of the four datasets we consider. On the other hand, Pseudo MLM, which uses both gold and pseudo monolingual data for MLM adaptation, consistently and significantly improves over Gold MLM for all tasks. Again, using Pseudo Trans-train on top of Pseudo MLM does not help and actually leads to relatively large performance loss for the syntactic tasks, such as POS tagging and Parsing.

Label Distillation brings significant improvements for the two syntactic tasks. Notably, it is the best performing method for POS tagging, but it still lags behind Pseudo MLM for Parsing. This is likely because Parsing is a much harder task than POS tagging to generate correct labels. The effect of Label Distillation on the NER task is less consistent—it improves over Pseudo Trans-train for WikiNER but not for MasakhaNER. This is because the named entity tags of the same words in different languages likely remain the same so that the pseudo task data probably has less noise for Label Distillation to have consistent benefits.

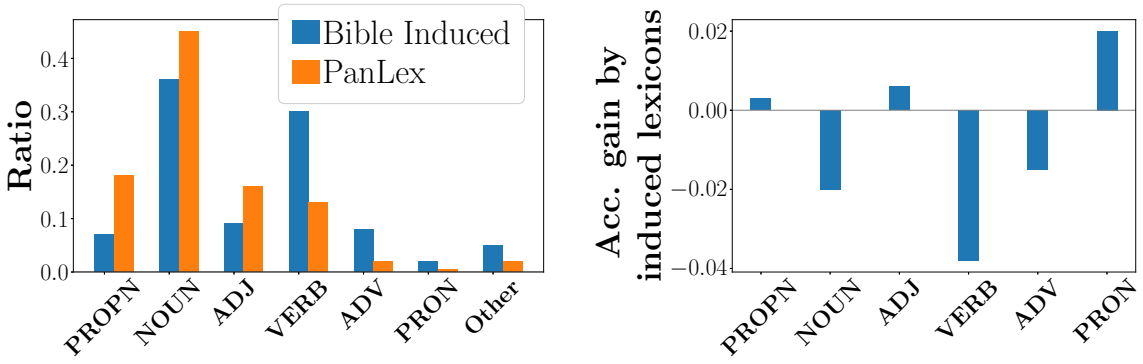

Figure 9.4: *left*: Ratio of words with different POS tags in each lexicon. *right*: POS accuracy gain of test words with different POS tags by using induced lexicons. The induced lexicons have more verbs but lead to worse performance on verbs.

Adding Induced Lexicons We examine the effect of using the lexicons augmented by word pairs induced from the Bible parallel data. The results can be found in Tab. 9.3. Adding the induced lexicon significantly improves the NER performance, while it hurts the two syntactic tasks.

To understand what might have prevented the syntactic tasks from benefiting from the extra lexicon entries, we plot the distribution of the part-of-speech tags of the words in PanLex lexicons and the lexicons induced from the Bible in Fig. 9.4. PanLex lexicons have more nouns than the Bible lexicons while the Bible lexicons cover more verbs than PanLex. However, the higher verb coverage in induced lexicons actually leads to a larger prediction accuracy drop for verbs in the POS tagging task. We hypothesize that the pseudo monolingual data created using the induced lexicons would contain more target language verbs with the wrong word order, which could be more harmful for syntactic tasks than tasks that are less sensitive to word order such as NER.

Discrepancies between the two NER datasets While WikiNER, along with POS tagging and Parsing, benefit the most from Pseudo MLM for Few-Text, MasakhaNER achieves the best result with Pseudo Trans-train. One possible explanation is that MasakhaNER contains data from the news domain, while WikiNER is created from Wikipedia. The pseudo monolingual data used for MLM is created from English Wikipedia articles, which could benefit WikiNER much more than MasakhaNER. On the other hand, the English NER training data for MasakhaNER is from the news domain, which potentially makes Pseudo Trans-train a stronger method for adapting the model simultaneously to the target language and to the news domain. One advantage of Pseudo MLM is that the English monolingual data is much cheaper to acquire, while Pseudo Trans-train is constrained by the amount of labeled data for a task. We show in **??** that Pseudo MLM has more benefit for MasakhaNER when we use a subset of the NER training data.

	bam	glv	mlt	myv
Gold MLM (Ours)	59.7	64.1	58.5	70.6
Ebrahimi and Kann (2021)	60.5	59.7	59.6	66.6
+Pseudo Trans-train	57.4	63.2	69.1	63.8
+Pseudo MLM	68.5	67.5	**72.3**	73.8
+Both	60.3	64.5	69.3	65.9
+Both(Label Distillation)	**69.4**	**68.8**	72.1	**74.3**

Table 9.4: Results for POS tagging with XLM-R. Our methods follow similar trend as on mBERT and they lead to significant gains compared to prior work.

9.7 Analyses

Performance with XLM-R We mainly use mBERT because it has competitive performance for under-represented languages and it is more computationally efficient due to the smaller size. Here we verify our methods have the same trend when used on a different model XLM-R (Conneau et al., 2019). We focus on a subset of languages in the POS tagging task for the Few-Text setting and the results are in Tab. 9.4. We use the smaller XLM-R base for efficiency, and compare to the best result in prior work, which uses XLM-R large (Ebrahimi and Kann, 2021). Tab. 9.4 shows that our baseline is comparable or better than prior work. Similar to the conclusion in § 9.6, Pseudo MLM is the competitive strategy that brings significant improvements over prior work. While adding Pseudo Trans-train to Pseudo MLM does not help, using Label Distillation further improves the performance.

Effect of Baseline Performance Using pseudo data might be especially effective for languages with lower performance. We plot the improvement of different languages over the baseline in Fig. 9.5, where languages are arranged with increasing baseline performance from left to right. We mainly plot Pseudo MLM and Pseudo Trans-train for simplicity. Fig. 9.5 shows that for both resource settings, lower performing languages on the left tend to have more performance improvement by using pseudo data.

Using NMT Model to Synthesize Data One problem with the pseudo data synthesized using word-to-word translation is that it cannot capture the correct word order or syntactic structure in the target language. If we have a good NMT system that translates English into the target language, we might be able to get more natural pseudo monolingual data by translating the English sentences to the target language.

Since the target languages we consider are usually not supported by popular translation services, we train our own NMT system by fine-tuning an open sourced many-to-many NMT model on the Bible

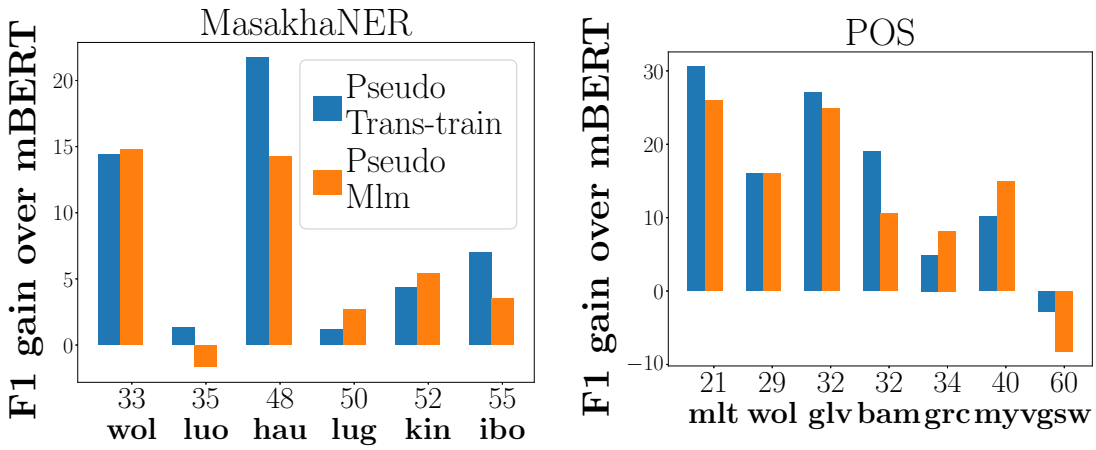

Figure 9.5: F1 gain over the baselines for languages with increasing baseline performance from left to right. Pseudo data tends to help more for languages with lower performance.

	WikiNER	MasakaNER	POS	Parsing
Lexicon	**45.0**	**56.0**	**63.7**	**40.7**
NMT	42.2	55.8	58.9	37.7

Table 9.5: F1 of using Pseudo MLM for Few-Text. Synthesizing data with NMT is consistently worse.

parallel data from English to the target language (details in ??). Instead of creating pseudo monolingual data using the lexicon, we can simply use the fine-tuned NMT model to translate English monolingual data into the target language.

The results of using NMT as opposed to lexicon for Pseudo MLM on all four tasks can be found in Tab. 9.5. Unfortunately, NMT is consistently worse than word-to-word translation using lexicons. We find that the translated monolingual data tend to have repeated words and phrases that are common in the Bible data, although the source sentence is from Wikipedia. This is because the NMT model overfits to the Bible data, and it fails to generate good translation for monolingual data from a different domain such as Wikipedia.

Comparison to Few-shot Learning Lauscher et al. (2020) found that using as few as 10 labeled examples in the target language can significantly outperform the zero-shot transfer baseline for languages included in mBERT. We focus on the zero-shot setting in this paper because the languages we consider have very limited data and it could be expensive or unrealistic to annotate data in every task for thousands of languages. Nonetheless, we experiment with k-shot learning to examine its performance on low-resource languages in the MasakhaNER task. Tab. 9.6 shows that using 10 labeled examples brings improvements over the mBERT baseline for a subset of the languages, and it is mostly worse than our

Method	hau	wol	lug	ibo	kin	luo
mBERT	48.7	33.9	50.9	55.2	52.4	35.3
Best Adapted	74.4	**60.3**	**61.6**	63.6	**63.8**	42.6
10-shot	44.5	49.1	52.7	56.2	51.2	46.2
100-shot	64.0	56.9	58.3	**65.5**	55.7	**51.6**
Best Adapt+100-shot	**76.1**	57.3	61.3	63.2	62.6	49.4

Table 9.6: Results on MasakhaNER for k-shot learning. We compare to the zero-shot mBERT baseline and our best adapted model.

best adapted model without using any labeled data. When we have access to 100 examples, few-shot learning begins to reach or exceed our zero-shot model. In general, few-shot learning seems to require more data to consistently perform well for under-represented languages while our adaptation methods bring consistent gains without any labeled data. Combining the best adapted model with few-shot learning leads to mixed results. More research is needed to understand the annotation cost and benefit of few-shot learning for low-resource languages.

9.8 Related Work

Several methods have been proposed to adapt pretrained language models to a target language. Most of them rely on MLM training using monolingual data in the target languages (Wang et al., 2020b; Chau et al., 2020; Muller et al., 2021; Ebrahimi and Kann, 2021; Pfeiffer et al., 2020b), competitive NMT systems trained on parallel data (Hu et al., 2020; Ponti et al., 2021), or some amount of labeled data in the target languages (Lauscher et al., 2020). These methods cannot be easily extended to low-resource languages with no or limited amount of monolingual data, which account for more than 80% of the World's languages (Joshi et al., 2020).

Bilingual lexicons have been commonly used for learning cross-lingual word embeddings (Mikolov et al., 2013; Ruder et al., 2019). Among these, some work uses lexicons to synthesize pseudo bilingual (Gouws and Søgaard, 2015; Duong et al., 2016) or pseudo multilingual corpora (Ammar et al., 2016). Mayhew et al. (2017) propose to synthesize task data for NER using bilingual lexicons. More recently, Khemchandani et al. (2021) synthesize monolingual data in Indian languages for adapting pretrained language models via MLM. Hu et al. (2021) argue that using bilingual lexicons for alignment hurts performance compared to word-level alignment based on parallel corpora. Such parallel corpora, however, are not available for truly under-represented languages. Reid and Artetxe (2021) employ a dictionary denoising objective where a word is replaced with its translation into a random language with a certain probability. This can be seen as text-to-text variant of our approach applied to multilingual pre-training. Akyurek and Andreas (2021) propose to use lexicons to improve the compositionality of NLP models.

None of the above works provide a systematic study of methods that utilize lexicons and limited data resources for adapting pretrained language models to languages with no or limited text.

9.9 Conclusion and Discussion

We propose a pipeline that leverages bilingual lexicons, an under-studied resource with much better language coverage than conventional data, to adapt pretrained multilingual models to under-represented languages. Through comprehensive studies, we find that using synthetic data can significantly boost the performance of these languages while the best method depends on the data availability. Our results show that we can make concrete progress towards including under-represented languages into the development of NLP systems by utilizing alternative data sources.

Our work also has some limitations. Since we focus on different methods of using lexicons, we restrict experiments to languages in Latin script and only use English as the source language for simplicity. Future work could explore the effect of using different source languages and combining transliteration (Muller et al., 2021) or vocabulary extension (Pfeiffer et al., 2021b) with lexicon-based data augmentation for languages in other scripts. We also did not test the data augmentation methods on higher-resourced languages as MLM fine-tuning and translate-train are already effective in that setting and our main goal is to support the languages with little textual data. Nonetheless, it would be interesting to examine whether our methods can deliver gains for high-resource languages, especially for test data in specialized domains.

We point to the following future directions: First, phrases instead of single word entries could be used to create pseudo data. Second, additional lexicons beyond PanLex could be leveraged. Some bilingual lexicons beyond PanLex are:

- Swadesh lists in about 200 languages in Wikipedia[6]
- Words in 3156 language varietities in CLICS[7]
- Intercontinental Dictionary Series in about 300 languages[8]
- 40-item wordlists in 5,000+ languages in ASJP[9]
- Austronesian Basic Vocabulary Database in 1,700+ languages[10]
- Diachronic Atlas of Comparative Linguistics in 500 languages[11]

Third, more effort could be spent on digitizing both existing monolingual data such as books Gref (2016) and lexicons into a format easily accessible by NLP practitioners. Although PanLex already covers over 5000 languages, some language varieties have only as little as 10 words in the database, while there exist many paper dictionaries that could be digitized through technologies such as OCR (Rijhwani et al.,

[6] https://en.wiktionary.org/wiki/Appendix:Swadesh_lists
[7] https://clics.clld.org/
[8] https://ids.clld.org/
[9] https://asjp.clld.org/
[10] https://abvd.shh.mpg.de/austronesian/
[11] https://diacl.ht.lu.se/

2020).[12] Lexicon collection is also relatively fast, which could be a more cost effective strategy to significantly boost the performance of many languages without lexicons. Finally, the quality of synthetic data could be improved by incorporating morphology. However, we find that there is virtually no existing morphological analysis data or toolkits for the languages we consider. Future work could aim to improve the morphological analysis of these low-resource languages.

[12] https://panlex.org/source-list/ contains a list of undigitized dictionaries.

Chapter 10

Conclusion

The adoption of technologies such as mobile phones and the Internet has made our planet more connected than ever before. Progress in multilingual NLP models can assist more seamless communication between speakers of different languages. However, as new users start to use multilingual NLP systems, these systems also have new challenges to serve speakers of unseen languages or dialects. While language technologies have advanced significantly for a handful of languages, the majority of language varieties on earth are falling behind. We hope that this thesis can contribute to building competitive NLP systems for people from different cultural and social backgrounds.

10.1 Summary of Contributions

The goal of the thesis is to create better NLP systems for low-resource languages by efficiently utilizing training data. In particular, we propose that multilingual learning methods should aim to utilize the commonalities between languages while recognizing the individual characteristics of each language.

Our contributions follow three general steps of model development. Before training any multilingual NLP model, selecting good training data often has a significant impact on any model training that follows. Part I explores data selection methods for multilingual training. We examine a heuristic method to select the most relevant data for multilingual training, and propose a general framework that leads to automatic data selection algorithms for multilingual data. After we select the suitable multilingual data for a specific objective, the next challenge is to properly represent the data in the neural network model. In Part II we design a novel word representation model for multilingual NMT that incorporates helpful inductive bias for cross-lingual transfer. Additionally, we also develop an efficient training algorithm to regularize word embeddings of multilingual pretrained models. After we train a good multilingual model that supports many different languages, it is often helpful to adapt the model to a specific language. In Part III, we propose methods that adapt pretrained multilingual models to two types of low-resource languages that are particularly challenging due to the paucity of data. We propose a novel model ensemble algorithm to support unseen language varieties at test time. We also experiment with various data augmentation methods for languages with no or limited textual data.

10.2 Future Directions

In this section, we outline some possible extensions of the works introduced in this thesis and future directions for multilingual NLP in general.

10.2.1 Efficient data selection

Our works in Part II show that good data selection and balancing strategies can have a significant impact on the performance of multilingual models. We propose a meta-learning framework for general data selection that shows promising results for multilingual training. One difficulty of our approach is that meta-learning requires longer training time and memory cost, which limits the application of our methods on large-scale models. Memory and computationally efficient meta-learning algorithms could be a promising direction to expand the practical impact of our framework. Furthermore, innovations in deep learning frameworks (Bradbury et al., 2018) and hardware architecture could potentially make meta-learning more efficient to use in practice.

10.2.2 Character-aware multilingual text representation

As discussed in Part I, the standard subword segmentation and representation methods are suboptimal for multilingual data. While the works in this thesis provide several improvements to the standard word representation through modeling and training regularization, more research is needed to design suitable multilingual text representation methods that are both computationally efficient and beneficial for cross-lingual transfer. Character-level representation can encourage cross-lingual transfer because it can potentially represent words with similar spellings from different languages to closer subspace than subword representations (Cherry et al., 2018a; Clark et al., 2022; Xue et al., 2021a). On the other hand, fully character based models often suffer from higher computational cost, and their performance is more sensitive to the choice of model architectures and other hyperparameters (Cherry et al., 2018b). Intuitively, text representations that utilize subword or word boundaries add helpful inductive biases to the model, while fully character-based representation might put additional burden of learning character compositions of the inputs on the rest of the model.

Given the strengths and weaknesses of both character and word/subword based text representations, it could be a promising future direction to combine the two so that we can maximize the benefits of both methods or have more flexible control over the trade-offs. Our work in Chapter 6 is an example of such methods, although the model incurs a higher engineering and computational cost. While character-based text representation is relatively popular and easy to implement for encoders, it is much more challenging to use for decoders because longer character sequences lead to slower inference speed. Future work could consider adding character level information as a regularization strategy at training time while keeping the subword based model architecture.

10.2.3 Multilingual benchmarks

Constructing an effective evaluation dataset and framework is usually the first step toward building any machine learning model for a specific task or a certain group of users. Although the NLP research field has developed various benchmarks for a wide range of tasks such as answering questions, dialogue, summarizing, common sense reasoning, etc., most works focus only on curating datasets in English (Sanh et al., 2021). Since the imbalance in the construction of data sets can amplify the existing gap in research efforts between English and other languages in the world, it is important to construct multilingual datasets that have wide language and task coverage. However, this is a nontrivial task because there are thousands of languages in the world and each language variety has unique linguistic and cultural attributes that need to be considered during the data collection process.

There are several potential directions for building more inclusive multilingual benchmarks. We can pay more attention to building websites and crowd-sourcing tools that allow native speakers of different languages to annotate or contribute data in their languages. This approach has already shown success in the speech processing community. For example, CommonVoice is a popular multilingual speech dataset where people speaking different languages can easily record a few voice clips to contribute to the dataset (Ardila et al., 2020). To further assist efficient data collection, we should also understand what types of language data are most helpful for model development and the amount of data that one needs to collect to achieve certain model performance.

10.2.4 Alternative data sources

Although there have been more work to improve NLP models for languages other than English, more fine-grained language varieties, such as regional dialects, have received very little attention in the research community. One of the biggest challenges with these language varieties is that they often have extremely limited or no textual training data at all. To close the gap, in Part III of the thesis we explored several methods addressing these challenges through test-time adaptation and data augmentation. One promising research direction to improve the performance of these fine-grained language varieties is to utilize alternative data sources. For example, many dialects are often used in casual conversational settings, so there are more audio or video recordings of these dialects than written text.

Some recent progress in speech pretraining opens up interesting opportunities to better serve speakers of fine-grained language varieties (Conneau et al., 2020; Babu et al., 2021a; Bapna et al., 2022). For example, we can use our proposed adaptation methods in Chapter 8 to dynamically adapt these multimodal pre-trained models to an unseen dialect at test time. It is also an important direction to use both speech and text data for multilingual pretraining because the learning signals and languages of each modality could be complementary. Moreover, speech and text data are more likely to come from different domains, so we should also design better data usage schedules or model architectures to utilize the domain information.

10.2.5 Personalized model adaptation

One of the core values of building multilingual NLP systems is to recognize and celebrate the uniqueness of native speakers of all languages in the world. More broadly speaking, each person speaks a language in a slightly different way and an intelligent NLP model should be able to adapt to the unique language variety spoken by each user. Personalized model adaptation has been studied in the context of NMT and dialogue (Michel and Neubig, 2018; Zheng et al., 2019), but it is still a relatively understudied area in NLP research. The speech processing community has more works on speaker adaptation for automatic speech recognition (Saon et al., 2013; Sarı et al., 2020; Zhao et al., 2021), probably because there are more individualized variations in speech utterances. As speech processing and NLP become more interconnected for real world applications such as speech translation (Zhang et al., 2021b; Lee et al., 2022) and conversational dialogue systems (Faruqui and Hakkani-Tür, 2022), personalized NLP models have the potential to bring significant positive impact on users with different cultural backgrounds.

Bibliography

Judit Ács. 2019. Exploring bert's vocabulary. 7.3

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Anuoluwapo Aremu, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. Masakhaner: Named entity recognition for african languages. In *TACL*. 9.4.1

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *NAACL*. 2.4.2, 2.4.3, 2.4.3, 5.4

Alham Fikri Aji, Genta Indra Winata, Fajri Koto, Samuel Cahyawijaya, Ade Romadhony, Rahmad Mahendra, Kemal Kurniawan, David Moeljadi, Radityo Eko Prasojo, Timothy Baldwin, Jey Han Lau, and Sebastian Ruder. 2022. One country, 700+ languages: NLP challenges for underrepresented languages and dialects in Indonesia. In *ACL*, Dublin, Ireland. Association for Computational Linguistics. 2.1

Ekin Akyurek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *ACL*, Online. Association for Computational Linguistics. 9.8

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*. 9.8

G. Anandalingam and Terry L. Friesz. 1992. Hierarchical optimization: An introduction. *Annals OR*. 4.5

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong

Kong, China. Association for Computational Linguistics. 4

R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. 2020. Common voice: A massively-multilingual speech corpus. In *LREC*, pages 4211–4215. 10.2.3

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. In *arxiv*. 2.4.3, 2.4.3, 5.1, 5.2, 5.2, 5.4, 5.6.2, 5.6.4, 5.8

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the Cross-lingual Transferability of Monolingual Representations. In *ACL*. 7.1, 7.5.1

Duygu Ataman and Marcello Federico. 2018. Compositional representation of morphologically-rich input for neural machine translation. *ACL*. 2.4.3, **??**, 6.6, 7.7

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *EMNLP*. 3.1, 4.1, 4.5

Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021a. Xls-r: Self-supervised cross-lingual speech representation learning at scale. 10.2.4

Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021b. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv*, abs/2111.09296. 2.4

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*. 2.2, 4.4.1, 6.2.1

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *EMNLP*. 2.4.3, 2.4.3

Ankur Bapna, Colin Cherry, Yu Zhang, Ye Jia, Melvin Johnson, Yong Cheng, Simran Khanuja, Jason Riesa, and Alexis Conneau. 2022. mslam: Massively multilingual joint pre-training for speech and text. 10.2.4

Atilim Gunes Baydin, Robert Cornish, David Martínez-Rubio, Mark Schmidt, and Frank Wood. 2018. Online learning rate adaptation with hypergradient descent. In *ICLR*. 4.2.3, 4.5, 5.8

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*. 4.1, 4.5

Steven Bird. 2020. Decolonising speech and language technology. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3504–3519, Barcelona, Spain (Online). International Committee on Computational Linguistics. 9.1

Graeme Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. In *COLING*, pages 3112–3122, Santa Fe, New Mexico, USA. Association for Computational Linguistics. 2.4.3

Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *ACL*. 1, 2.4.2, 9.1

Brenda Boerger. 2017. Rapid word collection, dictionary production, and community well-being. 9.1

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of EMNLP*. 7.1, 7.5.3

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs. 10.2.1

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. 1, 2.3, 2.4.3

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020a. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *COLING*. 8.1

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020b. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *COLING*. 9.1

Daniel Chandler. 2007. *Semiotic: The Basics*. 6.3

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of EMNLP 2020*. 2.4.3, 7.3, 9.1, 9.8

Ethan C. Chau and Noah A. Smith. 2021. Specializing multilingual language models: An empirical study. In *Workshop on MRL*, Punta Cana, Dominican Republic. Association for Computational Linguistics. 2.4.3

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *WMT*. 3.1

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*. 5.8

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018a. Revisiting character-based neural machine translation with capacity and compression. *CoRR*. 2.4.3, 6.2.2, 6.6, 7.7, 10.2.2

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018b. Revisiting

character-based neural machine translation with capacity and compression. In *EMNLP*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics. 10.2.2

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*, Ann Arbor, Michigan. Association for Computational Linguistics. 2.2

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *EMNLP*. 2.4.3, 7.3, 7.7

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Arxiv*. 2.2

Christopher Cieri, Mike Maxwell, Stephanie Strassel, and Jennifer Tracey. 2016. Selection criteria for low resource language programs. In *LREC*, Portorož, Slovenia. European Language Resources Association (ELRA). 2.1

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011a. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL*. 3.1, 6.4

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *TACL*, 8:454–470. 1.1

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011b. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL*. 3

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. In *NAACL*. 2.4.3, 10.2.2

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *EMNLP*. 7.1, 2, 7.7

Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals OR*, 153(1). 4.2.3, 4.5, 5.3

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition. 10.2.4

Alexis Conneau, Ankur Bapna, Yu Zhang, Min Ma, Patrick von Platen, Anton Lozhkov, Colin Cherry, Ye Jia, Clara Rivera, Mihir Kale, et al. 2022. Xtreme-s: Evaluating cross-lingual speech representations. *arXiv preprint arXiv:2203.10752*. 2.4.2

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. In *EMNLP*. (document), 1.1, 2.3, 2.4, 2.4.1, 2.4.3, 5.1, 5.2, 5.2, 5.6.4, 5.8, 7.1, 7.1, 7.2.1, 8.1, 9.1, 9.7

Alexis Conneau and Guillaume Lample. 2019. Crosslingual language model pretraining. In *NeurIPS*. 1.1, 2.3, 7.1, 8.1, 9.1

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *EMNLP*, pages 2475–2485. 7.5.1

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *WMT*. 3.3.1

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *ACL*. 2.3

Dorottya Demszky, Devyani Sharma, Jonathan H. Clark, Vinodkumar Prabhakaran, and Jacob Eisenstein. 2021. Learning to recognize dialect features. In *NAACL*. 8.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*. (document), 1, 1.1, 2.3, 2.4, 7.1, 7.1, 7.2.1, 8.1, 8.2, 8.4, 9.1, 9.1

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL*. 2.4.1, 2.4.3

Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. 2018. Adapting auxiliary losses using gradient similarity. *CoRR*, abs/1812.02224. 4.1, 4.2.2

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *ACL*. 3.1

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. *arXiv preprint arXiv:1606.09403*. 9.8

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. *NAACL*. 6.4.4

Abteen Ebrahimi and Katharina Kann. 2021. How to adapt your pretrained multilingual model to 1600 languages. In *ACL*, Online. Association for Computational Linguistics. 9.1, 2, 9.2.1, 9.4.1, 9.6.1, 9.6.2, **??**, 9.7, 9.8

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*. 1

Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2018a. Learning what data to learn. 4.4.2

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018b. Learning to teach. In *ICLR*. 4.1, 4.2.2, 4.5

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *EMNLP*, pages 595–605. 4.1, 4.2.2, 4.5

Manaal Faruqui and Dilek Hakkani-Tür. 2022. Revisiting the boundary between asr and nlu in the age of conversational dialog systems. In *Computational Linguistics*. 10.2.5

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation

of deep networks. In *ICML*. 4.1, 5.8

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. *NAACL*. 2.4.3, 3.1

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*, Austin, Texas. Association for Computational Linguistics. 2.4.1

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *EMNLP*. 4.5

Jost Gippert, Nikolaus Himmelmann, Ulrike Mosel, et al. 2006. *Essentials of language documentation*. Mouton de Gruyter Berlín. 9.1

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL*, pages 1386–1390, Denver, Colorado. Association for Computational Linguistics. 9.1, 9.8

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *ICML*. 4.5

Emily Kennedy Gref. 2016. *Publishing in North American Indigenous Languages*. Ph.D. thesis, University of London. 9.9

Algirdas Julien Greimas. 1983. *Structural semantics: An attempt at a method*. University of Nebraska Press. 6.3

Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky. 2007. Probabilistic and bottle-neck features for lvcsr of meetings. In *ICASSP*, volume 4, pages IV–757. IEEE. 4.2.4

Sophie Groenwold, Lily Ou, Aesha Parekh, Samhita Honnavalli, Sharon Levy, Diba Mirza, and William Yang Wang. 2020. Investigating African-American Vernacular English in transformer-based text generation. In *EMNLP*, Online. Association for Computational Linguistics. 1

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018a. Universal neural machine translation for extremely low resource languages. *NAACL*. 3.1, 6.1, 6.2.3, 6.2.4, **??**, 6.4.2, 6.4.3, 3, **??**, 6.6

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018b. Meta-learning for low-resource neural machine translation. In *EMNLP*. 5.8

Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *ICLR*. 2.4.3

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, Online. 9.2.1

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *Arxiv*. 2.4.3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recog-

nition. In *CPVR*. 4.4.1

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *ACL*, pages 3042–3051, Online. Association for Computational Linguistics. 2.4.3, 7.7

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *ACL*. 8.3

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*. 7.4

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural Computations*. 2.2

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *ICML*. 8.1

Jeremy Howard and Sebastian Ruder. 2018a. Universal language model fine-tuning for text classification. In *ACL*, Melbourne, Australia. Association for Computational Linguistics. 2.3

Jeremy Howard and Sebastian Ruder. 2018b. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*. 9.2.1

Junjie Hu, Melvin Johnson, Orhan Firat, Aditya Siddhant, and Graham Neubig. 2021. Explicit Alignment Objectives for Multilingual Bidirectional Encoders. In *NAACL*. 9.1, 9.8

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *ICML*. 2.4.1, 2.4.2, 7.1, 7.5.1, **??**, **??**, 7.5.3, 8.1, 8.4, 8.4, 9.1, 9.2.1, 9.4.1, 9.8

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhoun. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *EMNLP*. 7.1, 8.1

W. John Hutchins. 2000. Warren weaver and the launching of mt. In *Early Years in Machine Translation: Memoirs and biographies of pioneers*. 2.2

W. John Hutchins. 2004. The georgetown-ibm experiment demonstrated in january 1954. In *AMTA*. 2.2

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*. 4.1, 4.5, 4.5

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-paced curriculum learning. In *AAAI*. 4.4.1

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. 4.1, 4.2.2, 4.4.1

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016a.

Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*. **??**

Melvin Johnson et al. 2016b. Google's multilingual neural machine translation system: Enabling zero-shot translation. In *TACL*. 2.4.1, 2.4.1, 2.4.3, 5.1, 5.2, 5.6.1

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *ACL*, Online. Association for Computational Linguistics. 1, 2.1, 9.1, 9.2.2, 9.4.1, 9.8

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *Arxiv*. 2.4.3, 6.6, 7.7

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. 1

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *ACL*, Online. 2.4.3

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*. 5.8

Yash Khemchandani, Sarvesh Mehtani, Vaidehi Patil, Abhijeet Awasthi, Partha Talukdar, and Sunita Sarawagi. 2021. Exploiting language relatedness for low web-resource language model adaptation: An Indic languages study. In *ACL*, Online. Association for Computational Linguistics. 9.1, 9.8

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. 2.4.3, 7.7

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. *AAAI*. 2.4.3, 6.6, 7.7

Katrin Kirchhoff and Jeff A. Bilmes. 2014. Submodularity for data selection in machine translation. In *EMNLP*. 4.5

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. 3.1

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. 2.2

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *WMT*. (document), 1

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *EMNLP*, Hong Kong, China. 9.4.1

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. 2019. Do better imagenet models transfer better? In *CVPR*. 4.4.1

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report. 4.4.1

Taku Kudo. 2018a. Subword regularization: Improving neural network translation models with multiple

subword candidates. *ACL*. 6.1, 6.2.1

Taku Kudo. 2018b. Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*. 7.1, 7.2.2, 7.2.2, 7.2.3, 7.5.2

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*. 2.4.3, 3.3.2, 5.6.2, 7.1, 7.2.1

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Thang Luong, and Orhan Firat. 2021. Exploring routing strategies for multilingual mixture-of-experts models. In *OpenReview*, Online. 2.4.3

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. 2019. Reinforcement learning based curriculum optimization for neural machine translation. In *NAACL*, pages 2054–2061. 4.1, 4.2.2, 4.5

Sachin Kumar, Antonios Anastasopoulos, Shuly Wintner, and Yulia Tsvetkov. 2021. Machine translation into low-resource language varieties. In *ACL*. 8.1

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *EMNLP*, Online. Association for Computational Linguistics. 9.7, 9.8

Ann Lee, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Sravya Popuri, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, Juan Pino, and Wei-Ning Hsu. 2022. Direct speech-to-speech translation with discrete units. In *ACL*. 10.2.5

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL*. 2.4.3, 6.2.1, **??**, 6.6, 7.7

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, Online. Association for Computational Linguistics. 2.3

Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020b. MLQA: Evaluating Cross-lingual Extractive Question Answering. In *ACL*. 7.5.1

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Arxiv*. 2.4.3

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *ACL*. 2.4.3, 5.1, 5.8

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning language specific sub-network for multilingual machine translation. In *EMNLP*, pages 293–305, Online. Association for Computational Linguistics. 2.4.3

Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. 2018. Detecting and correcting for label shift with

black box predictors. *arXiv preprint arXiv:1802.03916*. 4.2.1

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019a. DARTS: differentiable architecture search. 4.1, 4.2.3, 4.5

Shikun Liu, Andrew J. Davison, and Edward Johns. 2019b. Self-supervised generalisation with meta auxiliary learning. *CoRR*, abs/1901.08933. 4.1

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*. 6.3.2

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Charbert: Character-aware pre-trained language model. In *COLING*. 2.4.3, 7.1, 7.7

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. In *arXiv*. 2.1

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *EMNLP*. 5.4

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *EMNLP*, Copenhagen, Denmark. Association for Computational Linguistics. 9.1, 9.8

Arya D. McCarthy, Rachel Wicks, Dylan Lewis, Aaron Mueller, Winston Wu, Oliver Adams, Garrett Nicolai, Matt Post, and David Yarowsky. 2020. The Johns Hopkins University Bible corpus: 1600+ tongues for typological exploration. In *LREC*, pages 2884–2892, Marseille, France. European Language Resources Association. 9.6.1

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *ACL*, pages 312–318, Melbourne, Australia. ACL. 10.2.5

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*. 9.8

David M. Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *EMNLP*. 2.4.1

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017a. Adversarial training methods for semi-supervised text classification. In *ICLR*. 7.7

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2017b. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 7.7

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL*. 3.1, 4.1, 4.5

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *ACL*, Online. Association for Computational Linguistics. 2.4.3, 9.8, 9.9

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. *EMNLP*. (document), 1.1, 2.4, 2.4.2, 2.4.3, 2.4.3, 3, 3.1, 3.3.1, 3.3.3, 4.3.2, 4.4.1, 5.1, 5.2, 6.1, 6.4.1, 2, **??**, 6.6

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V. Le, and Ruoming Pang. 2019. Domain adaptive transfer learning with specialist models. In *CVPR*. 4.5, 4.5

Toan Q. Nguyen and David Chiang. 2018. Transfer learning across low-resource, related languages for neural machine translation. In *NAACL*. 3.1, 6.6

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.2. 8.4, 9.4.1

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *EACL*. 5.4

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL: Demonstrations*. 5.6.2

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017a. Cross-lingual name tagging and linking for 282 languages. In *ACL*, pages 1946–1958. 1, 7.5.1, 8.4

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017b. Cross-lingual name tagging and linking for 282 languages. In *ACL*, pages 1946–1958, Vancouver, Canada. ACL. 9.4.1

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*. 4.4.1, 5.6.2

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*. Association for Computational Linguistics. 2.3

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*, Online. 8.4, 8.4

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *EMNLP System Demonstration*, Online. 8.2, 8.4

Jonas Pfeiffer, Ivan Vuli, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. In *Proceedings of EMNLP 2020*. 2.4.3, 2, 8, 8.1, 8.2, 8.4, 8.4, 9.1, 9.2.1, 9.4.1, 9.8

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. In *Proceedings of EMNLP 2021*. 9.1, 9.9

Minh Quang Pham, Josep Crego, Jean Senellart, and François Yvon. 2018. Fixing translation divergences in parallel corpora for neural MT. In *EMNLP*. 4.5

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *ACL*,

Florence, Italy. 7.1, 9.1

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*. 2.4.1

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *NAACL*. 4.5

Edoardo Maria Ponti, Julia Kreutzer, Ivan Vulić, and Siva Reddy. 2021. Modelling latent translations for cross-lingual transfer. In *Arxiv*. 9.8

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*. 5.6.2

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *ACL*. 7.1, 7.2.2, 7.2.2, 7.2.3, 7.5.2

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *ACL*. 9.4.2

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *NAACL*. 3.1, 3.3.1, 4.4.1, 5.6.1, 6.1, 6.4.1, 6.6

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. 1, 2.3

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67. 2.3

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *ACL*, pages 151–164, Florence, Italy. Association for Computational Linguistics. 9.4.1

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *NeurIPS*. 8.1

Machel Reid and Mikel Artetxe. 2021. PARADISE: Exploiting Parallel Data for Multilingual Sequence-to-Sequence Pretraining. *arXiv preprint arXiv:2108.01887*. 9.8

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *ICML*. 4.1, 4.2.3, 4.4.1, 4.5, 4.5, 5.8

Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. 2020. OCR Post Correction for Endangered Language Texts. In *EMNLP*, Online. Association for Computational Linguistics. 9.9

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards More Challenging and Nuanced Multilingual Evaluation. In *Proceedings of EMNLP 2021*. 9.2.1

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*, Melbourne, Australia. Association for Computational Linguistics. 2.3

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631. 9.1, 9.8

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*. 4.4.1

Unn Røyneland and Bård Uri Jensen. 2020. Dialect acquisition and migration in norway – questions of authenticity, belonging and legitimacy. In *Journal of Multilingual and Multicultural Development*. 8.1

Mohammad Salameh, Houda Bouamor, and Nizar Habash. 2018. Fine-grained Arabic dialect identification. In *COLING*. 8.1

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. 10.2.3

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. 2013. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. 10.2.5

Leda Sarı, Niko Moritz, Takaaki Hori, and Jonathan Le Roux. 2020. Unsupervised speaker adaptation using attention-based speaker memory for end-to-end asr. In *ICASSP*. 10.2.5

Tanja Schultz and Alex Waibel. 1998. Multilingual and crosslingual speech recognition. In *Proc. DARPA Workshop on Broadcast News Transcription and Understanding*. Citeseer. 2.4.1

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *ICASSP*. 7.2.1

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *ACL*. 2.3, 3.2.2

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL*. 2.4.3, 3.3.2, 6.1, 6.2.1, 7.1, 7.2.1

Rico Sennrich and Biao Zhang. 2019a. Revisiting low-resource neural machine translation: A case study. In *ACL*. (document), 1

Rico Sennrich and Biao Zhang. 2019b. Revisiting low-resource neural machine translation: A case study. In *ACL*. 4.4.1

C.E. Shannon. 1948. A mathematical theory of communication. In *Bell system technical journal*. 8.3

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff

Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *arXiv*. 2.4.3

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *EMNLP*. 2.4.1

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244. 4.2.1

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaroa. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. 2.4.3

Anil Kumar Singh. 2008. Natural language processing for less privileged languages: Where do we come from? where are we going? In *IJCNLP-08 Workshop on NLP for Less Privileged Languages*. 2.1

Sunit Sivasankaran, Emmanuel Vincent, and Irina Illina. 2017. Discriminative importance weighting of augmented training data for acoustic model training. In *ICASSP*. 4.1, 4.5

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *NAACL*. 4.5

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*. 4.2.1

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. 3.3.2, 6.1, 6.2.1, 6.2.2

Bente Ailin Svendsen and Unn Røyneland. 2008. Multiethnolectal facts and functions in oslo, norway. In *Internatioanl Journal of Bilingualism*. 8.1

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. In *CVPR*. 7.6.2

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. In *TACL*. 2.4.1

Samson Tan, Shafiq Joty, Lav Varshney, and Min-Yen Kan. 2020. Mind your inflections! Improving NLP for non-standard Englishes with Base-Inflection Encoding. In *EMNLP*, Online. Association for Computational Linguistics. 1

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*. 3.1

Sebastian Tschiatschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. 2014. Learning mixtures of submodular functions for image collection summarization. In *NIPS*. 4.5

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *ACL*. 4.1, 4.5

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adap-

tation for truly Universal Dependency parsing. In *EMNLP*. 2.4.3, 8.1

L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*. 6.5.6

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017a. Dynamic data selection for neural machine translation. In *EMNLP*. 3.1

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017b. Dynamic data selection for neural machine translation. In *EMNLP*. 4.4.3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. (document), 1, 2.2, 2.2, 4.4.1, 5.6.2, 6.3.2

Yogarshi Vyas, Xing Niu, and Marine Carpuat. 2018. Identifying semantic divergences in parallel text without annotations. In *NAACL*. 4.5

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2021a. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*. 8.3

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017. Sentence embedding for neural machine translation domain adaptation. In *ACL*. 3.1

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *EMNLP*. 4.1, 4.5

Wei Wang, Isaac Caswell, and Ciprian Chelba. 2019a. Dynamically composing domain-data selection with clean-data selection by "co-curricular learning" for neural machine translation. In *ACL*. 4.4.3

Xinyi Wang and Graham Neubig. 2019. Target conditioned sampling: Optimizing data selection for multilingual neural machine translation. In *ACL*. 1.3, 5.1, 5.8

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019b. Multilingual neural machine translation with soft decoupled encoding. In *ICLR*. 1.3, 4.4.1, 5.8, 7.7

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018a. Switchout: an efficient data augmentation algorithm for neural machine translation. In *EMNLP*. 3.2.3

Xinyi Wang, Hieu Pham, Paul Mitchel, Antonis Anastasopoulos, Jaime Carbonell, and Graham Neubig. 2019c. Optimizing data usage via differentiable rewards. In *arxiv*. 1.3, 5.4, 3, 5.9, 6.4.4

Xinyi Wang, Sebastian Ruder, and Graham Neubig. 2021b. Multi-view subword regularization. In *NAACL*. 1.3

Xinyi Wang, Sebastian Ruder, and Graham Neubig. 2022. Expanding pretrained models to thousands more languages via lexicon-based adaptation. In *ACL*. 1.3

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020a. Balancing training for multilingual neural machine translation. In *ACL*. 1.3

Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021c. Efficient test time adapter

ensembling for low-resource language varieties. In *EMNLP: Findings*. 1.3

Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2018b. Three strategies to improve one-to-many multilingual translation. In *EMNLP*. 5.8

Yining Wang, Long Zhou, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2019d. A compact and language-sensitive multilingual translation method. In *ACL*, pages 1213–1223, Florence, Italy. Association for Computational Linguistics. 2.4.3

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020b. Extending multilingual BERT to low-resource languages. In *EMNLP-Findings*, Online. Association for Computational Linguistics. 2.4.3, 9.1, 9.8

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 2020c. On negative interference in multilingual models: Findings and a meta-learning treatment. In *EMNLP*, pages 4438–4450, Online. Association for Computational Linguistics. 2.4.3

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *EMNLP*. 6.1, 6.3.2

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*. 4.2.2, 5.3

Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *NAACL*. 4.1, 4.5

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *EMNLP*. 2.3, 2.4.1, 7.1, 9.1

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc Le. 2018. Unsupervised data augmentation for consistency training. In *EMNLP*. 7.1, 7.7

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. Vocabulary learning via optimal transport for neural machine translation. In *ACL*, pages 7361–7373, Online. Association for Computational Linguistics. 2.4.3

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. In *ACL*, Vancouver, Canada. 7.7

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021a. Byt5: Towards a token-free future with pre-trained byte-to-byte models. In *arXiv*. 2.4.3, 10.2.2

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021b. mT5: A massively multilingual pre-trained text-to-text transformer. In *NAACL*, Online. Association for Computational Linguistics. 2.3

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *EMNLP*, pages 3685–3690. 7.5.1

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question

answering. *ACL*. 6.6

Li Yuan, Francis E.H.Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2020. Revisit knowledge distillation: a teacher-free framework. In *CVPR*. 7.6.2

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *BMVC*. 4.4.1

Poorya Zaremoodi, Wray L. Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *ACL*. 5.8

Biao Zhang, Ankur Bapna, Rico Sennrich, and Orhan Firat. 2021a. Share or not? learning to schedule language-specific capacity for multilingual translation. In *ICLR*, Online. 2.4.3

Chen Zhang, Xu Tan, Yi Ren, Tao Qin, Kejun Zhang, and Tie-Yan Liu. 2021b. Uwspeech: Speech to speech translation for unwritten languages. In *AAAI*. 10.2.5

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*. 4.2.1

Dakun Zhang, Jungi Kim, Josep Crego, and Jean Senellart. 2016. Boosting neural machine translation. *Arxiv 1612.06138*. 4.5

Xinsong Zhang and Hang Li. 2020. Ambert: A pre-trained language model with multi-grained tokenization. In *arxiv*. 2.4.3, 7.1, 7.7

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. 2018. An empirical exploration of curriculum learning for neural machine translation. *Arxiv, 1811.00739*. 4.5

Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, and Bin Ma. 2021. A unified speaker adaptation approach for ASR. In *EMNLP*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 10.2.5

Yinhe Zheng, Guanyi Chen, Minlie Huang, Song Liu, and Xuan Zhu. 2019. Personalized dialogue generation with diversified traits. 10.2.5

Changfeng Zhu, Heng Yu, Shanbo Cheng, and Weihua Luo. 2020. Language-aware interlingua for multilingual neural machine translation. In *ACL*, pages 1650–1655, Online. Association for Computational Linguistics. 2.4.3

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low resource neural machine translation. In *EMNLP*. (document), 1.1, 2.4, 3.1, 4.3.2, 6.6